SLAC-PUB-1448
CS-445
June 1974

A RELATIVELY EFFICIENT ALGORITHM - FOR FINDING NEAREST NEIGHBORS*

Jerome H. Friedman
Stanford Linear Accelerator Center
Stanford, California

and

Forest Baskett and Leonard J. Shustek
Computer Science Department, Stanford University
Stanford, California

June 1974

ABSTRACT

An algorithm that finds the k nearest neighbors to a point, from a sample of size N in a d-dimensional space, with an expected number of distance calculations

$$E[n_d] \lesssim \frac{1}{\sqrt{\pi}} \left[kd\Gamma\left(\frac{d}{2}\right)\right]^{\frac{1}{d}} (2N)^{1-\frac{1}{d}}$$

is described, its properties examined and verified with simulated data. For bivariate normal data, an average of 24 distance calculations is required to find the nearest neighbor to a point from 1000 prototype samples.

(Submitted to IEEE Trans. on Computers)

*Work supported by U.S. Atomic Energy Commission under contract AT(043)515

INTRODUCTION

Nearest neighbors have been shown to be an important nonparametric technique for multivariate density estimation and pattern classification.[1-6] For classification, a sample of prototype feature vectors is drawn from each category, correctly labeled by an external source. For each test point to be classified, the set of k-closest prototype points (feature vectors) is found and the test point is assigned to that category having the largest representation in this set. For density estimation, the volume, V(k), containing the closest k-points to each of the N sample points, is used to estimate the local sparsity , $\hat{s}$, (inverse density) by $\hat{s} = NV(k)/ k$.

The application of these techniques has been severely limited by the computational resources required for finding the nearest neighbors. The feature vectors for the complete set of samples must be stored, and the distances to them calculated for each classification or density estimation. Several modifications to the k-nearest neighbor rule have been suggested that are computationally more tractable but whose statistical properties are unknown.[7-8] The condensed nearest-neighbor rule[9] mitigates both the storage and processing requirements by choosing a subset of the prototype vectors such that the nearest neighbor rule correctly classifies all of the prototypes.

Fisher and Patrick[10] suggest a preprocessing scheme for reducing the computational requirements of nearest neighbor classifications when the test sample is much larger than the prototype set. For this case, it is worthwhile to use considerable computation preprocessing the prototypes so that processing can be reduced for each test sample. Their technique orders the prototypes so that each point tends to be far away from its predecessors in the ordered list. By examining these prototype points in this order and

having precalculated distances between prototypes, the triangle inequality can be applied to eliminate distance calculations from the test vector to many of the prototypes. (All of the prototypes must be examined, however.) The algorithm is examined only for k=1 in two dimensions where for bivariate normal data a median number of approximately 58 distance calculations were required for 1000 prototypes, after preprocessing.

This paper describes a straightforward preprocessing technique for reducing the computation required for finding the k-nearest neighbors to a point from a sample of size N in a d-dimensional space. This procedure can be profitably applied to both density estimation and classification, even when the number of test points is considerably smaller than the number of prototypes. This preprocessing requires no distance calculations. (It can, however, require up to $dN\log_2 N$ comparisons.) The distance function (dissimilarity measure) is not required to satisfy the triangle inequality. With a Euclidean distance measure[11] the average number of prototypes that need be examined is

$$E[n_d] \lesssim \frac{1}{\sqrt{\pi}} \left[k\, d\, \Gamma\left(\frac{d}{2}\right)\right]^{\frac{1}{d}} (2N)^{1-\frac{1}{d}}, \qquad (1)$$

after preprocessing.

For the case of bivariate normal data with d=2, k=1, and N=1000, eqn (1) predicts an average of 36 distance calculations, whereas 24 are actually required. The performance of the algorithm is compared to eqn (1), with simulated data for several values of k, d, N, and underlying density distributions of the prototype sample points.

BASIC PROCEDURE

The preprocessing for this algorithm consists basically of ordering the prototype points on the values of one of the coordinates. For each test point, the prototypes are examined in the order of their projected distance from the test point on the sorted coordinate. When this projected distance becomes larger than the distance (in the full dimensionality) to the k closest point of those prototypes already examined, no more prototypes need be considered; the k-closest prototypes of the examined points are those for the complete set. Figure 1 illustrates this procedure for the nearest neighbor (k=1) in two dimensions.

A simple calculation gives an approximation to the expected number of prototypes that need be examined before the above stopping criteria is met. For simplicity, consider N prototypes uniformly distributed in a d-dimensional hypercube and a Euclidean distance measure. Assume also that N is large enough so that effects due to the boundaries are not important. For this case, the volume, v, of a d-dimensional sphere, centered at the test point containing k-prototypes, is a random variable distributed according to a beta distribution

$$p(v)dv = \frac{N!}{(k-1)!(N-k)!}\, v^{k-1}\,(1-v)^{n-k}\, dv \qquad 0 \le v \le 1. \tag{2}$$

The radius of this sphere, given by

$$r_d = \left[\frac{d\Gamma(\frac{d}{2})}{2\pi^{d/2}}\, v\right]^{\frac{1}{d}}, \tag{3}$$

is also a random variable. Let

$$s_d = \left[2\pi^{d/2} \,/\, d\Gamma\left(\frac{d}{2}\right)\right]^{\frac{1}{d}} .$$

Then $v = (s_d r_d)^d$ and the distribution for r_d becomes

$$p(r_d)dr_d = \frac{N!\, ds_d}{(k-1)!(N-k)!} (s_d r_d)^{dk-1} [1-(s_d r_d)^d]^{N-k} dr_d. \qquad (4)$$

The stopping criteria is met when the projected distance from the test point to a prototype along the sorted coordinate is greater than r_d. This projected distance is uniformly distributed. The expected fraction of prototypes, then, that must be examined is just twice the expected value of r_d given by eqn (4).[12] Various other statistics, such as the variance, median, and percentiles can also be calculated from eqn (4). These calculations must be done numerically since the integrals cannot be evaluated analytically.

A close upper bound[13] on $E[r_d]$ can be derived from eqn's (2) and (3) by

$$E[r_d] \lesssim \hat{r}_d = \left[\frac{d\Gamma(\frac{d}{2})}{2\pi^{d/2}} E[v]\right]^{\frac{1}{d}} \qquad (5)$$

where from eqn (2)

$$E[v] = \frac{k}{N+1} . \qquad (6)$$

The upper bound on the expected fraction of prototypes that must be examined is then $2\hat{r}_d$, and the upper bound on the expected number of prototypes, $E[n_d]$, is $2\hat{r}_d N$. Combining these results

$$E[n_d] \leq \hat{n}_d = 2 \left[\frac{d\Gamma(\frac{d}{2})}{2\pi^{d/2}} \frac{k}{N+1}\right]^{\frac{1}{d}} N. \qquad (7)$$

Simplifying this expression and approximating N+1 by N, one has the result shown in eqn (1). The variance of n_d is similarly approximated by

$$E[n_d - E(n_d)]^2 \lesssim 4 \left[\left(\frac{d\Gamma(\frac{d}{2})}{2\pi^{d/2}} \right)^2 \frac{k(N-k+1)}{(N+1)^2(N+2)} \right]^{\frac{1}{d}} N \qquad (8)$$

so that the coefficient of variation becomes (to the same approximation)

$$C(n_d) \lesssim \left[\frac{N-k+1}{k(N+2)} \right]^{1/(2d)} . \qquad (9)$$

Other statistics of the distribution can be similarly calculated.

These calculations all presuppose a uniform distribution of the prototype sample. This is seldom the case in application. For an arbitrary prototype density $p(\vec{x})$, the quantity

$$u_k(\vec{x}) = \int_{s_k(\vec{x})} p(\vec{x})\, d\vec{x} \qquad (10)$$

has the beta distribution of eqn (2), where $s_k(\vec{x})$ is the d-dimensional sphere centered at the test point $\vec{x}$, containing k prototypes. Also, the projected densities on the coordinate axes are no longer uniform.

FULL PROCEDURE

The nonuniformity of the axis projections can be used to advantage to increase the efficiency of the algorithm. For a given expected radius $E[r_d]$, the points that need to be considered are those that lie in the interval $\Delta x \simeq 2E[r_d]$, centered at the projected test point. That projection axis, for which the number of such prototypes is least, should be chosen for maximum efficiency. If the points are ordered only along one coordinate in the preprocessing (basic procedure), then the one with the smallest average projected density (largest spread) should be chosen. In the full procedure the points are ordered on several or all of the coordinates and the one with the smallest local projected density in the neighborhood of the test point is chosen. Thus, the ordering of the prototypes depends on the location of the test point in the feature space.

For each test point, the local projected sparsity on each axis is estimated as

$$s_i = \left| X_{i,p_i+n/2} - X_{i,p_i-n/2} \right| \tag{11}$$

where X_{ij} is the ith coordinate of the jth ordered prototype and p_i is the position of the test point in the ith projection. The ordering of the prototypes on the particular coordinate for which s_i is maximum, $j = \max_{1 \le i \le d}^{-1}\{s_i\}$, is chosen separately for each test point.

The number of prototypes, n, over which the sparsity is averaged on each projection should correspond to a distance of about $2E[r_d]$. For a uniform distribution, this is given approximately by eqn (5). The number of prototypes within this interval (again for a uniform distribution) is $E[n_d]$, given by eqn (1). For nonuniform distributions, both $E[r_d]$ and the various projected $E[n_d]$'s will be different. Since the density distribution of the prototypes is usually unknown, a reasonable approximation is to use the uniform distribution results,[14] that is $n=E[n_d]$ as given by eqn (1).

The starting list of the k-nearest prototypes are those whose position is closest to the test point in the jth coordinate. The other prototype points are then examined in order of their increasing projected distance from the test point until the stopping condition

$$d_k^2 \le (X_{j,p_j} - X_{j,\ell})^2 \tag{12}$$

is met for some point ℓ. Here d_k^2 is the distance squared to the kth nearest prototype of those examined up to that point. The current list of k closest prototypes is then correct for the entire sample.

The expected number of prototype points, $E_{full}[n_d]$, that need to be considered when applying this full procedure of choosing the optimum ordering coordinate individually for each test point, can be calculated using arguments

and assumptions similar to those that led to eqn (1). The result is

$$E_{full}[n_d] \simeq E\left[\frac{\left(\prod_{i=1}^{d} s_i\right)^{\frac{1}{d}}}{\max\limits_{1\le i\le d}\{s_i\}}\right] E[n_d] \qquad (13)$$

where the s_i's are the local projected sparsities on each of the coordinates near each test point, and $E[n_d]$ is the expected number if all projected sparsities were the same. For uniformly distributed prototypes, $E[n_d]$ is given by eqn (1) and the s_i's are distributed normally about their average values.

Actual values for $E_{full}[n_d]$ are difficult to calculate, but eqn (13) can be used to gain insight into the strategy's effect. For example, if the spread of one of the coordinates is a factor of R larger, on the average, than the others which all have approximately equal spread, then eqn (13) gives

$$E_{full}[n_d] \simeq \frac{1}{R^{1-\frac{1}{d}}} E[n_d] \; . \qquad (14)$$

Eqn (13) clearly shows that the strategy will always increase the efficiency of the algorithm and be most effective when the variation of the prototype density is greatest.

GENERAL DISSIMILARITY MEASURES

Although the above discussion has centered on the Euclidean distance

$$d(\vec{X}_m,\vec{X}_n) = \left[\sum_{i=1}^{d} (X_{im} - X_{in})^2\right]^{1/2} \qquad (15)$$

as a measure of dissimilarity between feature vectors, nowhere in the procedure is it required. In fact, the triangle inequality is not required. This technique can be applied with any dissimilarity measure

$$d(\vec{X}_m,\vec{X}_n) = g\left[\sum_{i=1}^{d} f_i(X_{im},X_{in})\right] \tag{16}$$

as long as the functions f and g satisfy the basic properties of symmetry

$$f(x,y) = f(y,x) \tag{17a}$$

and monotonicity

$$\begin{aligned} &g(x) \geq g(y) \qquad \text{if } x > y \\ &f(x,z) \geq f(x,y) \quad \begin{cases} \text{if } z \geq y \geq x \\ \text{or if } z \leq y \leq x \ . \end{cases} \end{aligned} \tag{17b}$$

The performance of the algorithm does depend upon the dissimilarity measure and, in particular, the result contained in eqn (1) applies only to the Euclidean metric. The dependence on k and N contained in eqn (1) is the same for any Minkowski p metric

$$d(\vec{X}_m,\vec{X}_n) = \left[\sum_{i=1}^{d} |X_{im} - X_{in}|^p\right]^{\frac{1}{p}} \qquad (p \geq 1) \tag{18}$$

since for these distance measures the volume of a d-dimensional "sphere" grows with radius, r, as $v \propto r^d$. Because of their computational advantage, the two most often used Minkowski metrics, besides the Euclidean metric (p = 2), are the city-block or taxi cab distance (p = 1) and the maximum coordinate distance, $d(\vec{X}_m,\vec{X}_n) = \max_{1\leq i\leq d}\{|X_{im} - X_{in}|\}$ $(p = \infty)$. Upper bounds on the average number of distance calculations, $E[n_d]$, analogous to eqn (1), can be derived in a similar manner for these distance measures. The results are

$$E_1[n_d] \lesssim (kd!)^{\frac{1}{d}} N^{1-\frac{1}{d}} \qquad (p = 1) \tag{1a}$$

$$E_\infty[n_d] \lesssim k^{\frac{1}{d}} N^{1-\frac{1}{d}} \qquad (p = \infty) \ . \tag{1b}$$

SIMULATION EXPERIMENTS

In order to gain insight into the performance of the algorithm and compare it to that predicted by eqn (1), several simulation experiments were performed. For each simulation N+1 random d-dimensional points were drawn from the appropriate probability density function. The number of distance calculations required to find the k-nearest neighbors to each point using the full procedure (sorting on all coordinates) was determined and then averaged over all of the points. This procedure was then repeated ten times with different random points from the same probability density function. The average of these ten trials was then taken as the result of the experiment, and the statistical uncertainty was taken to be $1/\sqrt{10}$ times the standard deviation about the mean for the ten trials. These uncertainties were all less than one percent and for the larger samples were around 0.1 percent.

These simulation results are presented in Figures 2-5 where the variation of the average number of distance calculations with N, k, d, underlying distribution, and distance measure is compared to the upper bound predicted by eqn (1) (solid lines). Figure 2 shows the dependence on N (d=2, k=1) for several underlying distributions. These distributions are uniform on the unit square, bivariate normal with unit dispersion matrix, and bivariate Cauchy

$$p(x,y) = \frac{1}{(1+x^2)(1+y^2)} \quad . \tag{19}$$

Figure 3 shows the dependence on k (d=2, n=100 and 1000) for uniform and normal data. Figure 4 shows the dependence on d (k=1, N=100 and 1000) again, for both uniformly and normally distributed data. Figure 5 shows the dependence on d (k=1, N=1000, uniform distribution) for several different Minkowski p metrics, namely p=1 (city block distance), p=2 (Euclidean distance), and p=∞ (maximum coordinate distance).

DISCUSSION

These simulation experiments show that eqns (1, 1a, 1b) do, indeed, provide a close upper bound on the average number of distance calculations required by the algorithm to find nearest neighbors. Although these formulae always slightly overestimate the actual number, they quite accurately reflect the variation with N, k, d and p. As predicted by eqn (13), the number of distance calculations tends to diminish for increasing density variation of the sample points.[15] It is interesting to note that for d=2 and k=1, this algorithm requires a smaller average number of distance calculations for 10,000 prototypes than does the brute force method (calculating all of the distances) for 100 prototypes.

The relative efficiency of this algorithm (as compared with the brute force method) decreases slightly with increasing k and more rapidly with increasing dimensionality d. In eight dimensions for 1000 prototypes (k=1) the average number of distance evaluations is reduced by approximately 40%. Although not dramatic, this is still quite profitable in terms of the preprocessing requirements.

As indicated by eqns (1, 1a, 1b) and verified in Figure 5, the growth of $E[n_d]$ with dimensionality depends strongly on the choice of the distance measure. For Minkowski p metrics, it is easy to show that $E_p[n_d]$ grows more slowly with d for increasing p. The results of eqn (1c) and Figure 5 indicate that if a distance measure is chosen on the basis of rapid calculation, the maximum coordinate distance ($p=\infty$) is the natural choice since it also minimizes the number of distance calculations, especially for high dimensionality.

A crude calculation gives a rough idea of how many test points, N_t, are required (in terms of the number of prototypes, N_p, k, and d) for the preprocessing procedure to be profitable. The preprocessing requires approximately

$dN_p \log_2 N_p$ compares, memory fetches and stores. Each distance calculation requires around d multiplies, subtractions, additions, and memory fetches. Assuming all of these operations require equal computation, then the preprocessing requires about $3dN_p \log N_p$ operations, while it saves approximately $4dN_t(N_p - E[n_d])$ operations. Thus, for the procedure to be profitable

$$4dN_t(N_p - E[n_d]) > 3dN_p \log N_p$$

or crudely[16]

$$N_t > N_0 \approx \frac{N_p \log_2 N_p}{N_p - E[n_d]} \quad . \tag{20}$$

$E[n_d]$ is given approximately by eqn (1) (Euclidean metric). For d=2, k=1, and 1000 prototypes, N_0 is around 10, whereas for d=8, k=1, and 1000 prototypes, one has $N_0 \approx 25$.

Although these results are quite crude, it is clear that the number of test points need not be large, compared to the number of prototypes, before the algorithm can be profitably applied. For density estimation, where $N_t = N_p = N$, the procedure is profitable so long as N_0/N is small compared to one.

The only adjustable parameter in this algorithm is the number of projection coordinates, m, on which the data are sorted. This parameter can range in value from one (basic procedure) to d (full procedure). If less than the full procedure is employed, then those axes with the largest spread should be chosen. Arguments similar to those that lead to eqn (14) can be used to estimate the efficiency for this case. For the case where all coordinates have approximately equal spread, eqn (13) can be used to show that the increase in efficiency, as additional sorted coordinates are added, is proprotional to 1/m. Results of simulations (not shown) verify this dependence.

The tendency toward decreasing relative efficiency with increasing dimensionality cannot be mitigated by requiring the distance measure to satisfy the triangle inequality

$$d(\vec{x}_m,\vec{x}_n) \geq \left| d(\vec{x}_m,\vec{x}_\ell) - d(\vec{x}_\ell,\vec{x}_n) \right| . \tag{21}$$

In this case distance calculations can be avoided for those prototypes, $\vec{x}_n$, for which

$$d(\vec{x}_t,\vec{x}_k) < \left| d(\vec{x}_t,\vec{x}_\ell) - d(\vec{x}_\ell,\vec{x}_n) \right| \tag{22}$$

where $\vec{x}_t$ is the test point, $\vec{x}_k$ is the k-th nearest prototype of those already examined and $\vec{x}_\ell$ is a prototype for which $d(\vec{x}_t,\vec{x}_\ell)$ and $d(\vec{x}_\ell,\vec{x}_n)$ have already been evaluated (and saved). The use of eqn (22) will be most effective when the dispersion of interpoint distances in the prototype sample is greatest. In this case, $d(\vec{x}_t,\vec{x}_k)$ will tend to be small whereas $d(\vec{x}_t,\vec{x}_\ell)$ and $d(\vec{x}_\ell,\vec{x}_n)$ will quite often be dissimilar, making the right hand side on eqn (22) large. Since distance varies as the d-th root of the volume, the distance variation will decrease with increasing dimensionality for a given density variation.

For a uniform density distribution in a d-dimensional space, the coefficient of variation of the interpoint distance is

$$\left[\frac{\langle r_d^2\rangle}{\langle r_d\rangle^2}\right]^{1/2} = \frac{1}{\sqrt{d(d+2)}} \tag{23}$$

which decreases as $1/d$ for increasing d. Thus, the usefulness of the triangle inequality is affected by the "curse of dimensionality" in the same manner as the algorithm discussed above.

The performance of this algorithm has been discussed in terms of the number of prototypes that need to be examined and distances calculated. Although this is closely related to the actual computing requirements, it should be kept in mind that the true performance also depends on the details of implementation and the hardware capabilities of a specific computer.

CONCLUSION

A simple algorithm has been presented for finding nearest neighbors with computation proportional to $k^{\frac{1}{d}} N^{1-\frac{1}{d}}$ and preprocessing proportional to dNlogN. The algorithm can be used with a general class of dissimilarity measures, not just those that satisfy the triangle inequality. The algorithm takes advantage of local variations in the structure of the data to increase efficiency. Formulas that enable one to calculate a close upper bound on the expected performance for common metrics have been derived. Simulation experiments have been presented that illustrate the degree to which these formulas bound the actual performance.

ACKNOWLEDGMENT

Helpful discussions with William H. Rogers, Sam Steppel, and John E. Zolnowsky are gratefully acknowledged.

FOOTNOTES AND REFERENCES

1. E. Fix and J.L. Hodges, Jr., "Discriminatory Analysis; Small Sample Performance," USAF School of Aviation Medicine, Randolf Field, Texas, Project 21-49-004, Report No. 11, August 1952.

2. D.O. Loftsgaarden and C.P. Quesenberry, "A Nonparametric Density Function," Ann. Math. Stat., Vol. 36, pp. 1049-1051, 1965.

3. T.M. Cover and P.E. Hart, "Nearest Neighbor Pattern Classification," IEEE Trans. Info. Theory, Vol. IT-13, pp. 21-27, 1967.

4. T.M. Cover, "Rates of convergence of nearest neighbor decision procedures," Proc. First Annual Hawaii Conference on Systems Theory, pp. 413-415, January 1968.

5. T.J. Wagner, "Convergence of the nearest neighbor rule," IEEE Trans. Info. Theory, Vol. IT-17, pp. 566-571, Sept. 1971.

6. K. Fukunaga and L.D. Hostetler, "Optimization of k-nearest-neighbor density estimates," IEEE Trans. Info. Theory, Vol. IT-19, pp. 320-326, May 1973.

7. C. Barns, "An easy mechanized scheme for an adaptive pattern recognizer," IEEE Trans. Elec. Comp., Vol. EC-15, 385-387, June 1966.

8. E.A. Patrick and F.P. Fisher, "Generalized k-nearest neighbor decision rule," Journal of Information and Control, Vol. 16, pp. 128-152, April 1970.

9. P.E. Hart, "The condensed nearest neighbor rule," IEEE Trans. Info. Theory, Vol. IT-14, pp. 515-516, May 1968.

10. F.P. Fisher and E.A. Patrick,"A preprocessing algorithm for nearest neighbor decision rules," Proc. Nat. Electronics Conf., Vol. 26, pp. 481-485, Dec. 1970.

11. Similar formulae for other distance measures are discussed below.

12. Prototypes must be examined for a projected distance r_d both above and below the test point position.

13. Taking the dth root of the average rather than the average of the dth root will cause a slight overestimation that decreases with increasing d. For d=2, this overestimation is around 10%, while for d=8 it is 6%.

14. Simulation results indicate that the performance of the algorithm is very insensitive to the choice of n.

15. The actual improvement, using the full procedure, is somewhat underrepresented. Simulations using the basic procedure (sorting on one coordinate only) gave considerably higher values of $E[n_d]$ (20% to 30%) for the nonuniform densities than for the uniform density. The full procedure results presented in Figures 2-4 show the nonuniform densities with smaller values for $E[n_d]$ than the uniform cases (again 20% to 30%). Thus, the improvement over the basic procedure for these nonuniform densities is around twice that indicated. The full procedure also reduces the coefficient of variation of n_d to about half that for the basic procedure and that predicted by eqn (9).

16. This calculation is extremely dependent on the specific computer upon which the algorithm is implemented, and the results of eqn (20) should be regarded as only a crude estimate.

FIGURE CAPTIONS

FIGURE 1. Illustration of the basic procedure for the nearest neighbor (k=1) in two dimensions.

FIGURE 2. Average number of distance calculations required to find the nearest neighbor (k=1) for bivariate uniform (□), normal (○), and Cauchy (◊) density distributions as a function of prototype sample size, N.

FIGURE 3. Average number of distance calculations required to find the k nearest neighbors with bivariate uniform (□) and normal (○) density for 100 and 1000 prototypes.

FIGURE 4. Average number of distance calculations required to find the nearest neighbor for uniform (□) and normal (○) density distributions with 100 and 1000 prototypes as a function of dimensionality, d.

FIGURE 5. Variation with dimensionality, d, of the average number of distance calculations required to find the nearest neighbor (k=1, N=1000, uniform distribution) for several Minkowski p metrics, p=1 (◊), p=2 (○), p=∞ (□).

FIGURE 1

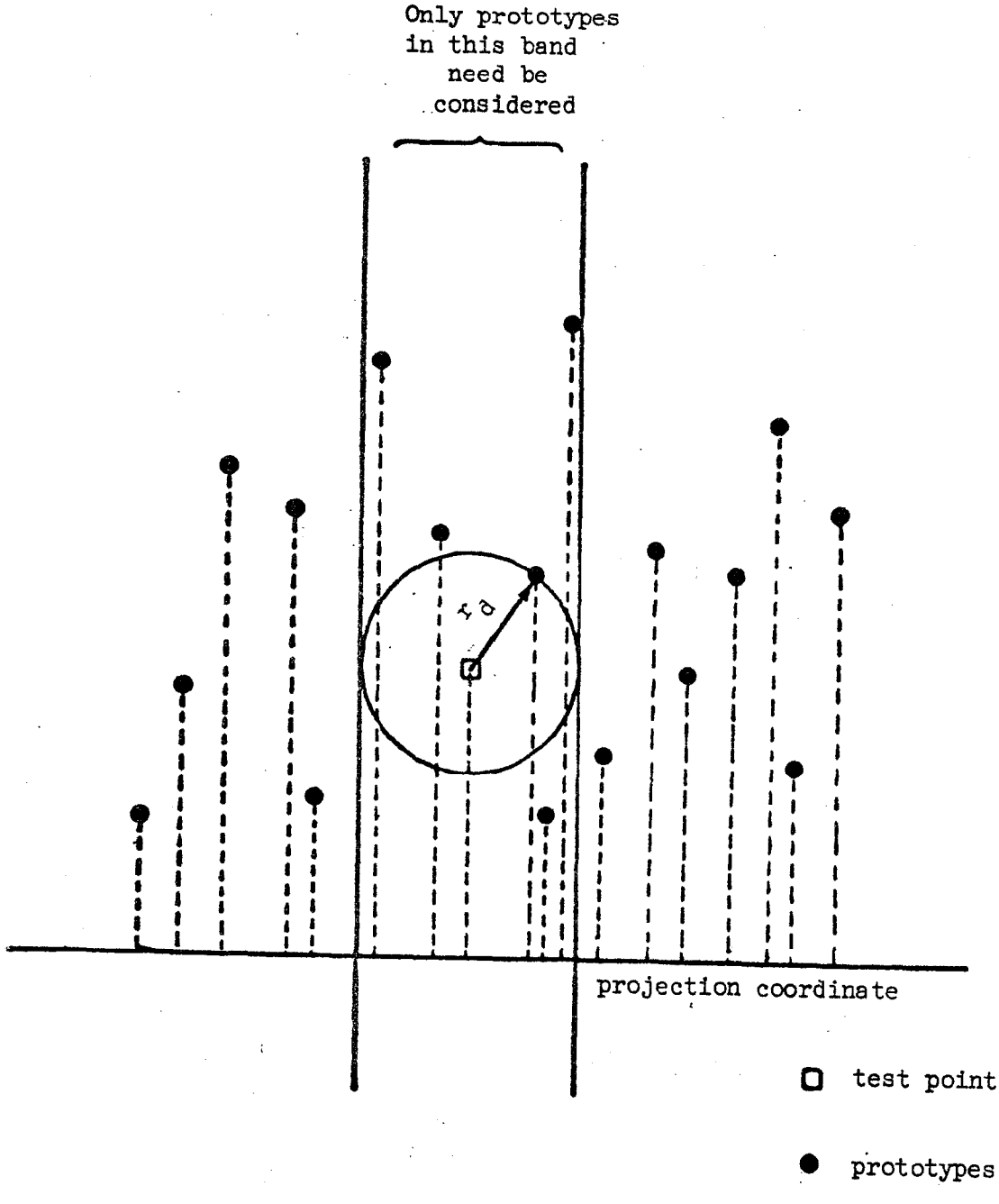

Only prototypes
in this band
need be
considered
projection coordinate
test point
prototypes

FIGURE 2

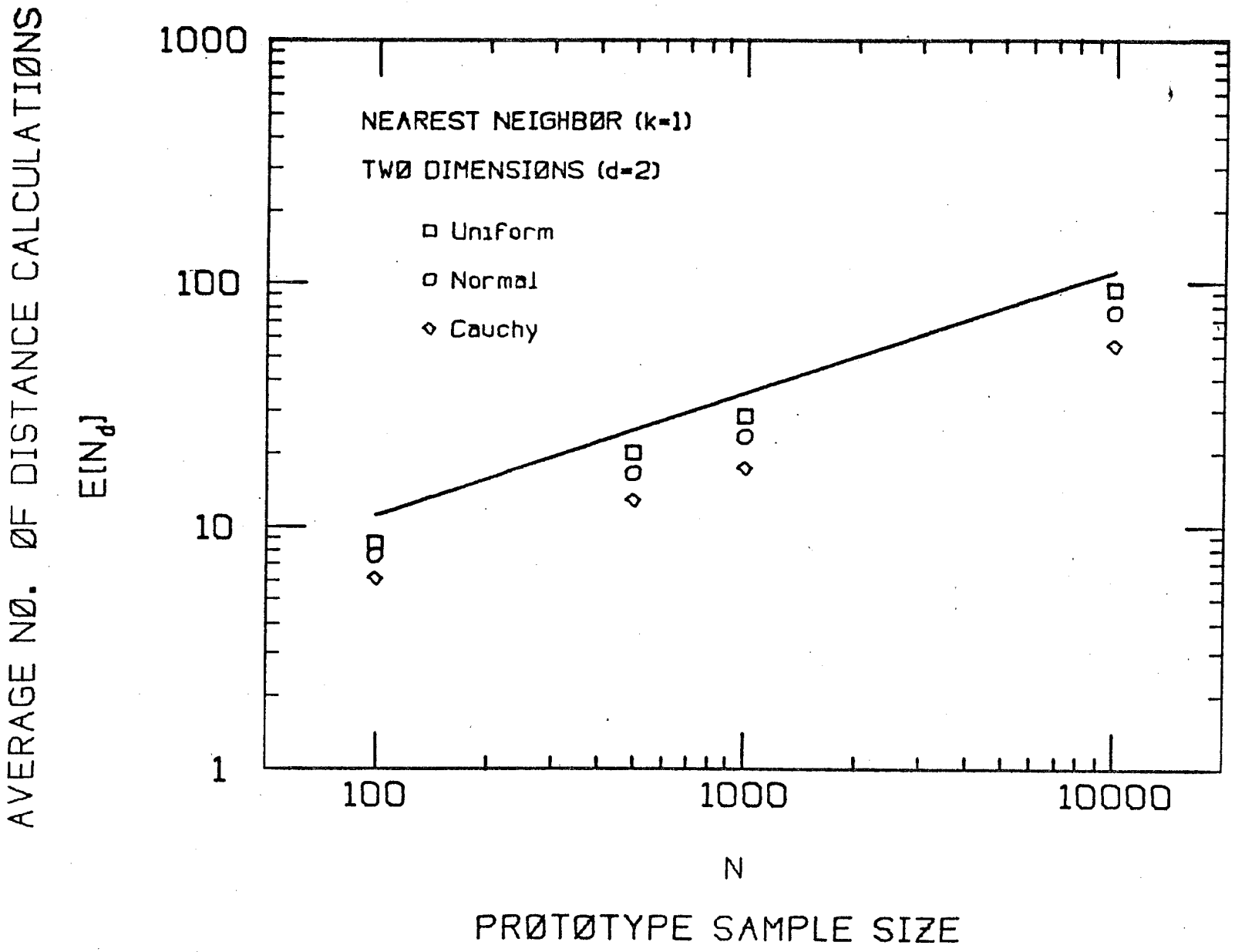

AVERAGE NØ. ØF DISTANCE CALCULATIØNS
E[N_d]
1000
100
10
1
NEAREST NEIGHBØR (k=1)
TWØ DIMENSIØNS (d=2)
Uniform
Normal
Cauchy
100
1000
10000
N
PRØTØTYPE SAMPLE SIZE

FIGURE 3

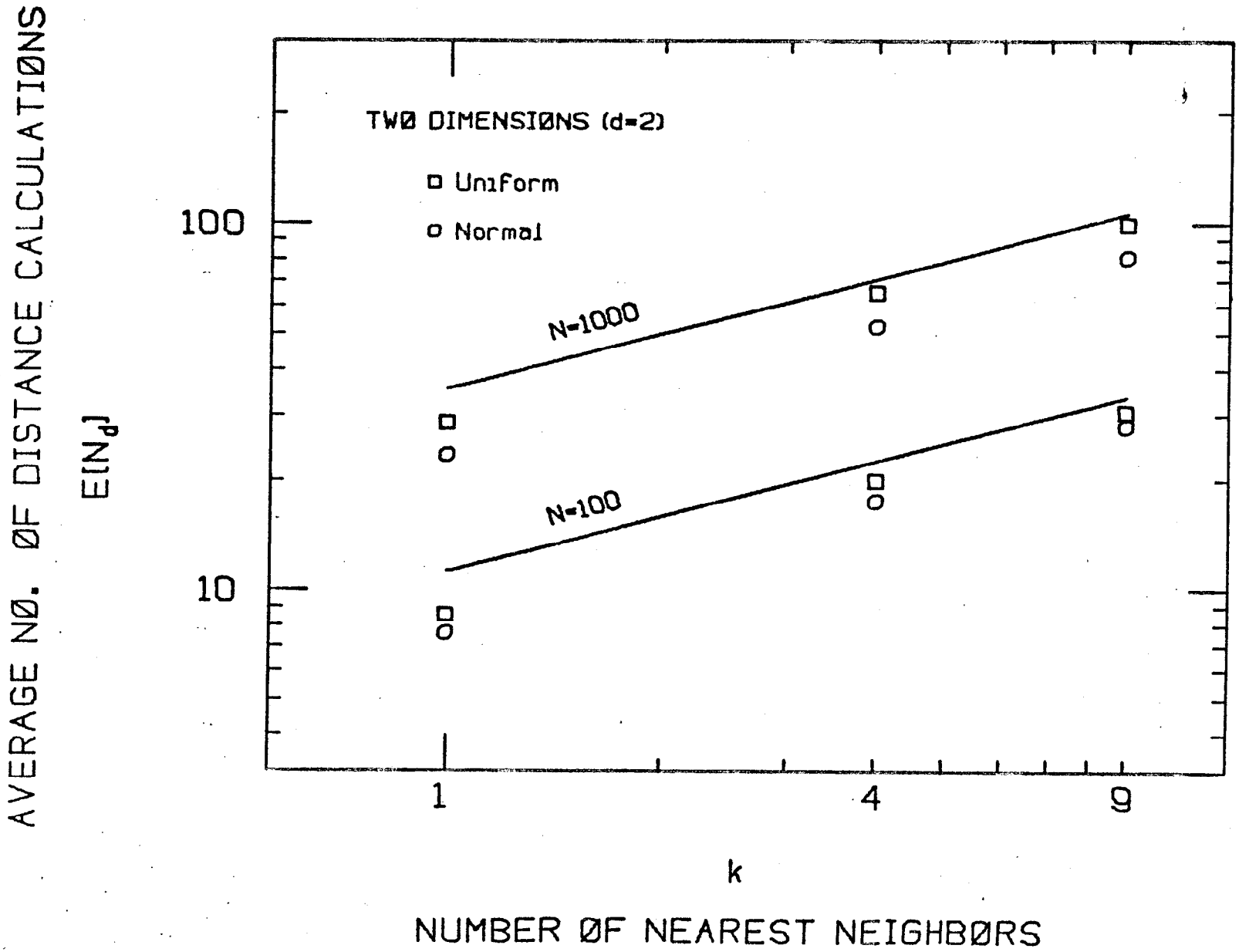

AVERAGE NØ. ØF DISTANCE CALCULATIØNS
E[N_d]
TWØ DIMENSIØNS (d=2)
Uniform
Normal
N=1000
N=100
100
10
1
4
9
k
NUMBER ØF NEAREST NEIGHBØRS

FIGURE 4

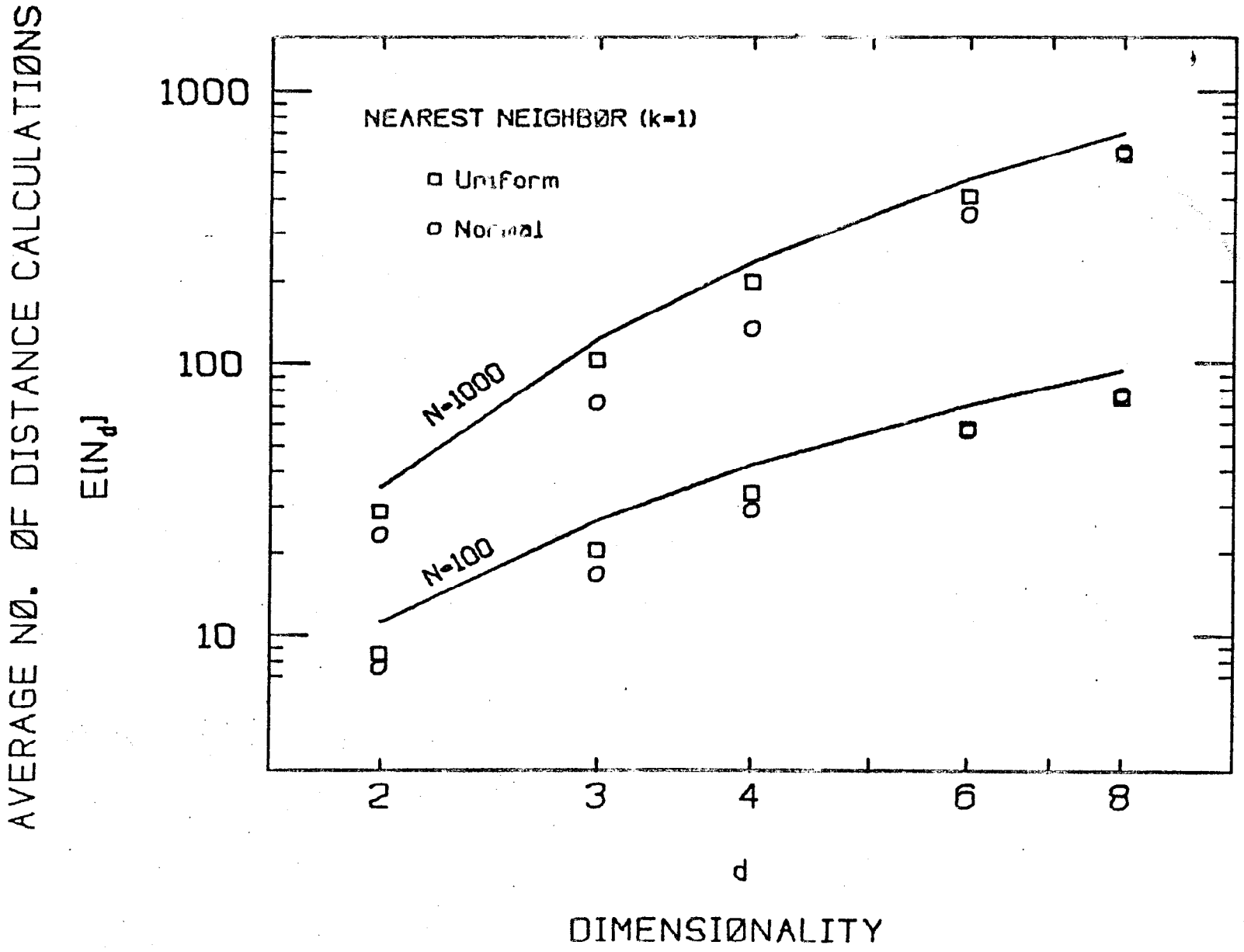
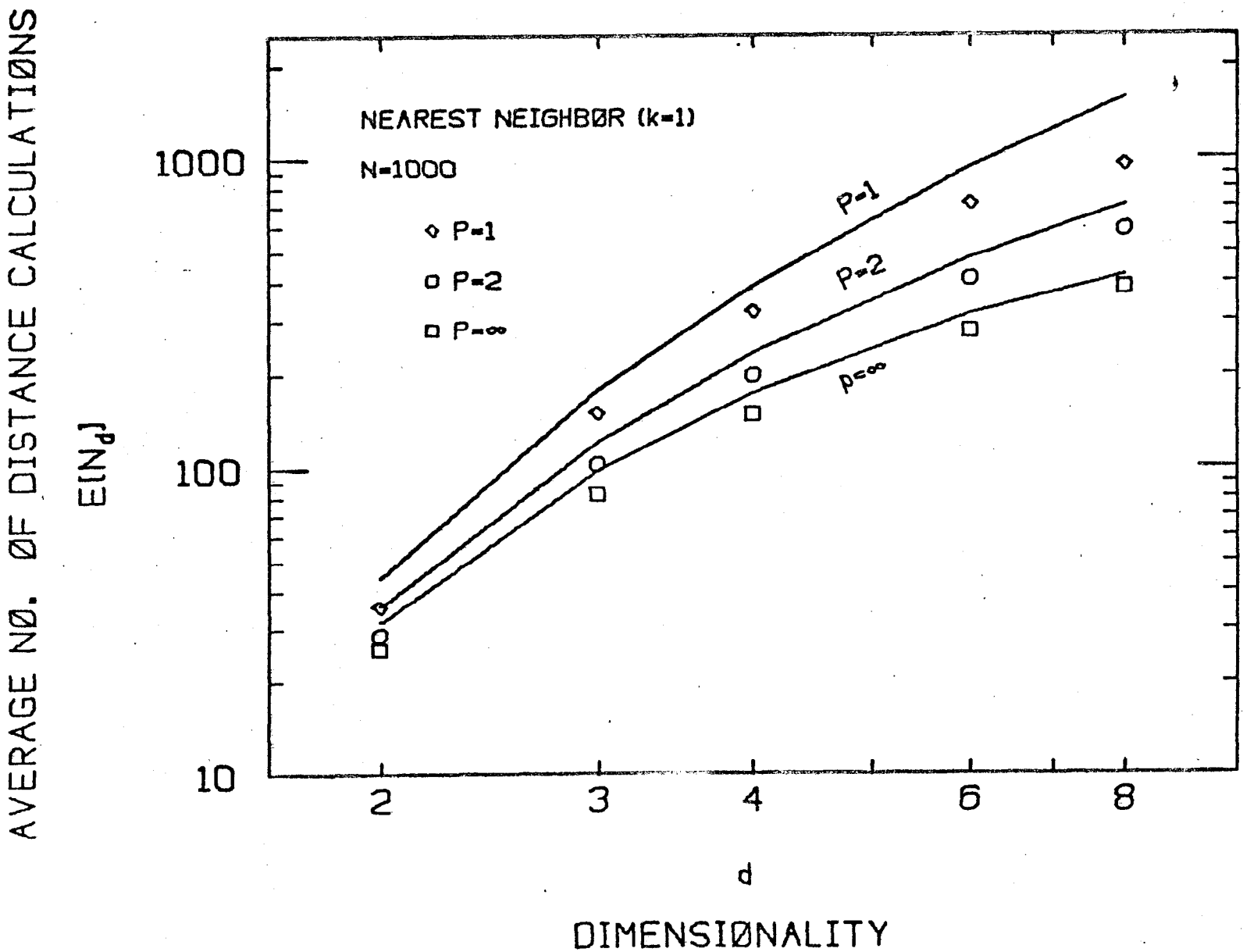

AVERAGE NØ. ØF DISTANCE CALCULATIØNS
E[N_d]
1000
100
10
NEAREST NEIGHBØR (k=1)
Uniform
Normal
N=1000
N=100
2
3
4
6
8
d
DIMENSIØNALITY

FIGURE 5


AVERAGE NØ. ØF DISTANCE CALCULATIØNS
E[N_d]
1000
100
10
NEAREST NEIGHBØR (k=1)
N=1000
P=1
P=2
P=∞
p=1
p=2
p=∞
2
3
4
6
8
d
DIMENSIØNALITY