Image Cover Sheet

CLASSIFICATION	SYSTEM NUMBER 131315
UNCLASSIFIED	

TITLE

NEAREST-NEIGHBOUR SEARCH IN CONSTANT TIME: AN ALTERNATIVE TO NEURAL NETWORKS FOR PATTERN MATCHING APPLICATIONS

System Number:

Patron Number:

Requester:

Notes: Paper #2 contained in Parent Sysnum #129006

DSIS Use only:

Deliver to: DK

NEAREST-NEIGHBOUR SEARCH IN CONSTANT TIME: AN ALTERNATIVE TO NEURAL NETWORKS FOR PATTERN MATCHING APPLICATIONS

Pierre Zakarauskas and John M. Ozard

Defence Research Establishment Pacific, FMO Victoria, B.C. Canada, V0S 1B0.

Contact Pierre Zakarauskas, (604) 363-2879

ABSTRACT

// In this paper we present an alternate algorithm to the use of neural networks for pattern matching in continuous space, and present performance estimates as a function of the dimensionality of the search space. Neural networks divide decision space with hyperplanes whose positions are adjusted during training, usually through gradient descent. The decision regions thus obtained are not necessarily optimal for noisy data. Another drawback of neural networks is the rapid increase in training time with the size of the problem, leading to the quasi impossibility of training them for very large problems. In order to obtain robust pattern matching for problems where the number of patterns becomes very large, we propose the use of a fast nearest-neighbour algorithm. This algorithm finds the training pattern closest to a test pattern, in a time which is asymptotically constant with the total number of patterns. The algorithm first partitions the search space, and finds which partition each of the training pattern falls into. When a test pattern is supplied, one compares it to all patterns within the partitions which are within a given distance R to the test pattern. If the distance d between the closest pattern found during the first pass and the test pattern is less than R, then the search stops. If not, then a second pass is done using d as a search parameter to select partitions. This algorithm guarantees that the nearest-neighbour is found. The mean number of operations done to find the nearest-neighbour is presented as a function of the density of patterns per partition and the dimensionality of the search space. A global minimum is shown to exist for the mean number of operations at slightly less than one pattern per partition. //

CONTENT

Pattern matching and classification

Nearest-neighbour rule

Fast nearest-neighbour search
-Necessity of searching more than one bucket

Algorithms analyzed:

- Product partitioning
- k-d tree partitioning
- Ordered partitioning
  - Cylindric
  - Limited

Analysis of performance

Pattern matching and classification

- Neural networks often used for performing pattern matching tasks

- Given a distribution of labelled patterns x_q find class C_i that x_q is most likely to belong to.

- Multilayered feedforward neural networks divide parameter space with decision hyperplanes.
  - Efficient, but difficult to find correct planes
  - Scales extremely badly with size of problem
  - Classification not optimum in high noise situations

- Classic pattern matching techniques more robust, and maybe more economical than neural networks in certain cases.

Nearest-neighbour rule

Classification on the basis of the class most heavily represented among the k-nearest neighbors (kNN) to the test pattern is asymptotically Bayes' risk efficient.

$$\rho \leq r_{NN} \leq 2\rho$$

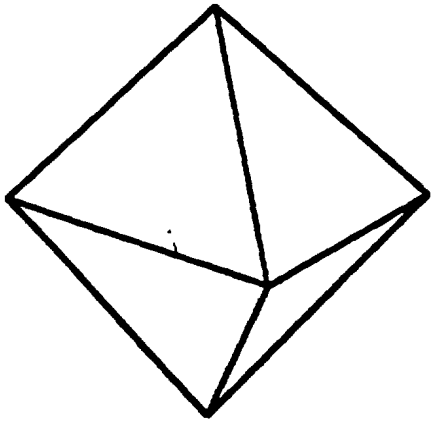
The probability of the NN of being in the same bucket as the test pattern is

$$p_{NN} = \left(1 + n^{-1/d}\right)^{-d}.$$

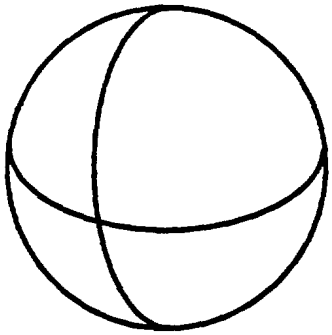
The Minkowski p-metric is measure of distance D_p between two points , and is defined as

$$D_p(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{1/p}$$

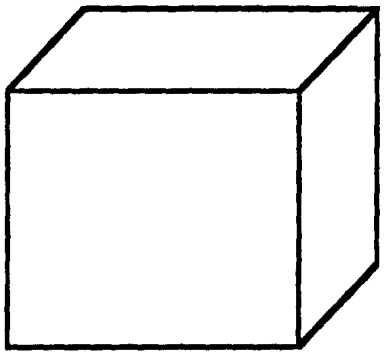
Shape of hypersurface for different Minkowski p-metrics



$p = 1$



$p = 2$



$p = \infty$

The hypervolume of the search p-sphere is

$$V_r = K_{p,d} r^d$$

where $K_{p,d}$ is a constant determined by p and d.

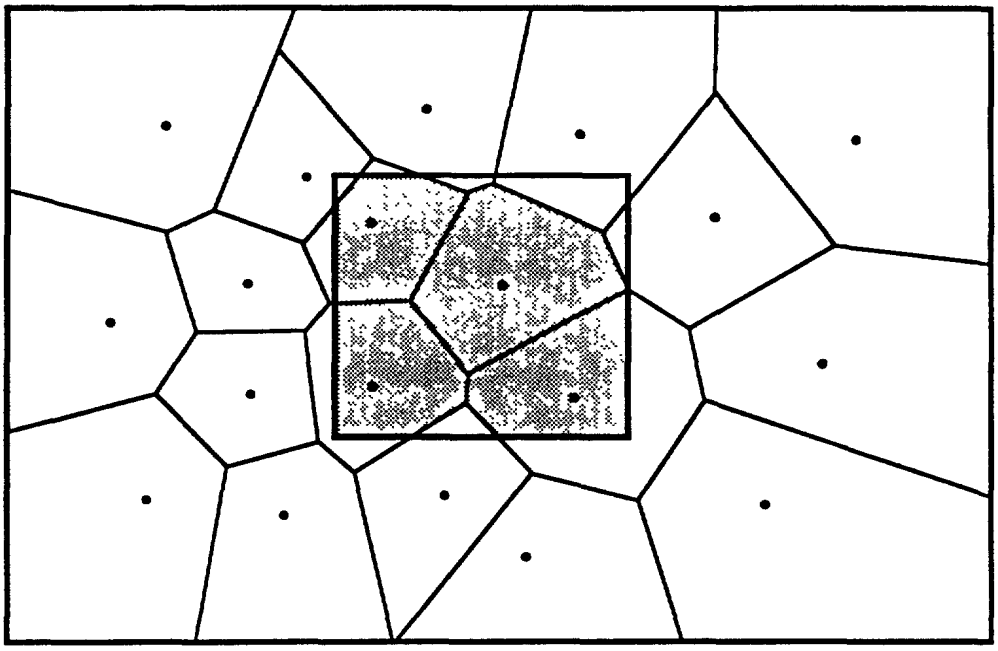
$$K_{1,d} = \frac{2^d}{d!}$$

$$K_{2,d} = \frac{\pi^{d/2}}{\Gamma\left(d/2 + 1\right)},$$

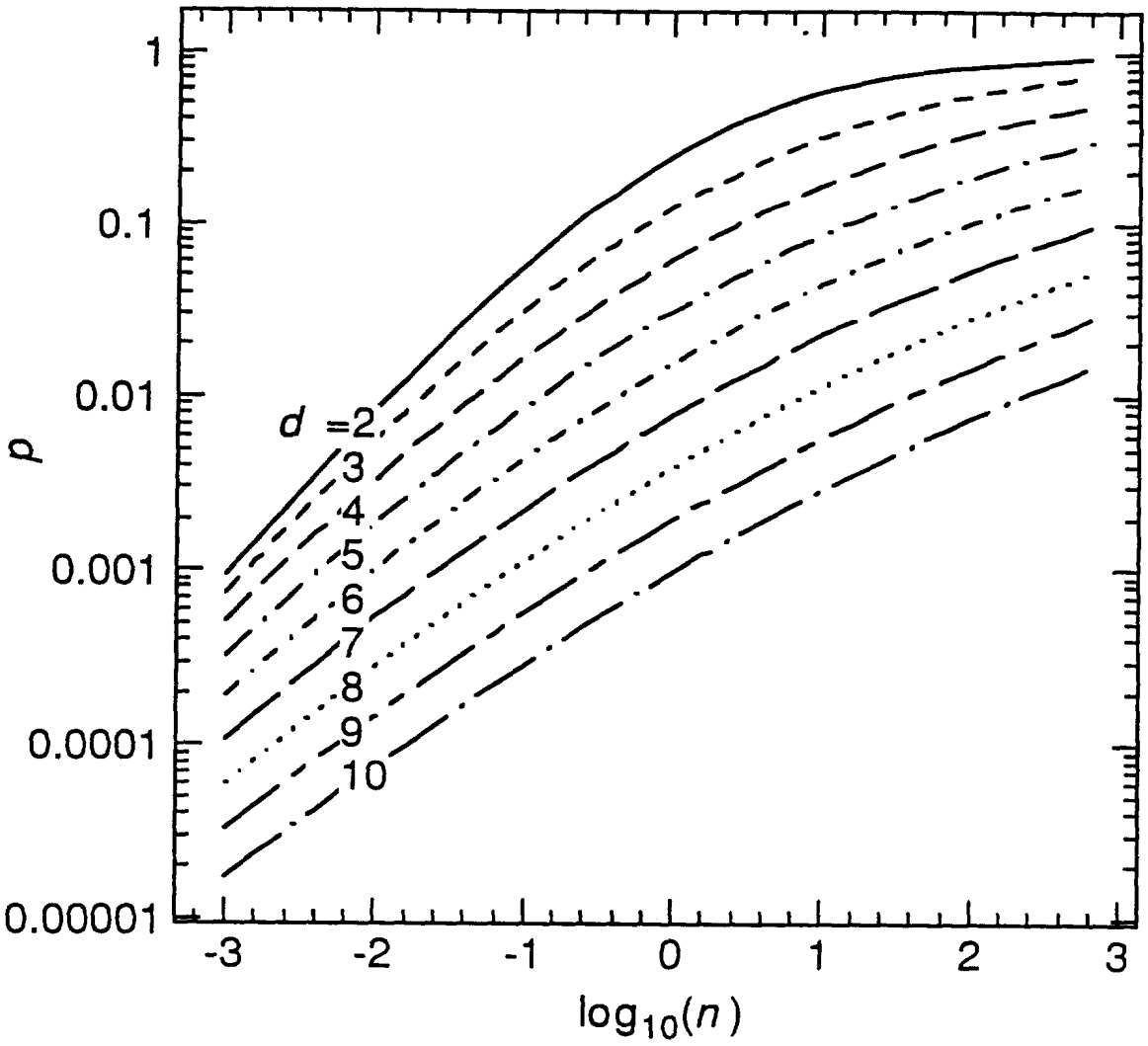
and

$$K_{\infty,d} = 2^d$$

Points having a nearest-neighbour within a given cube



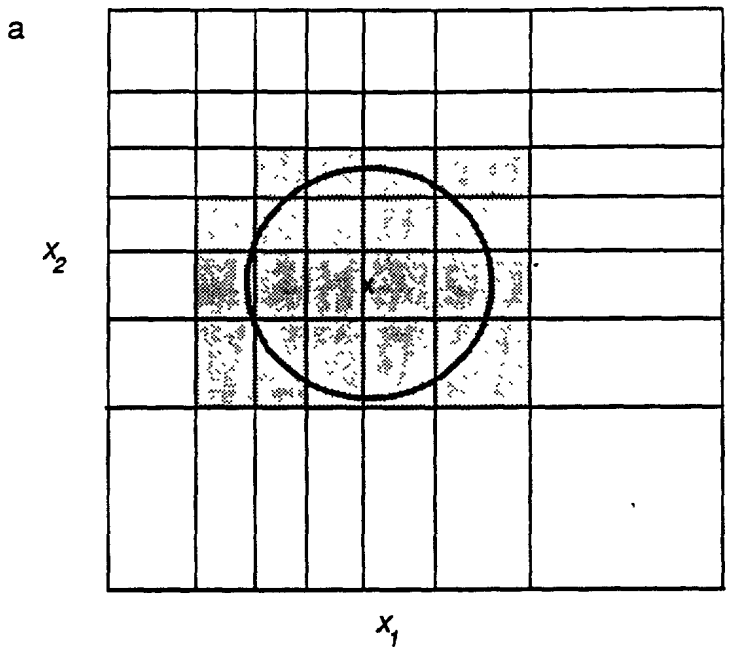
Probability of the NN being in the same bucket as the test pattern

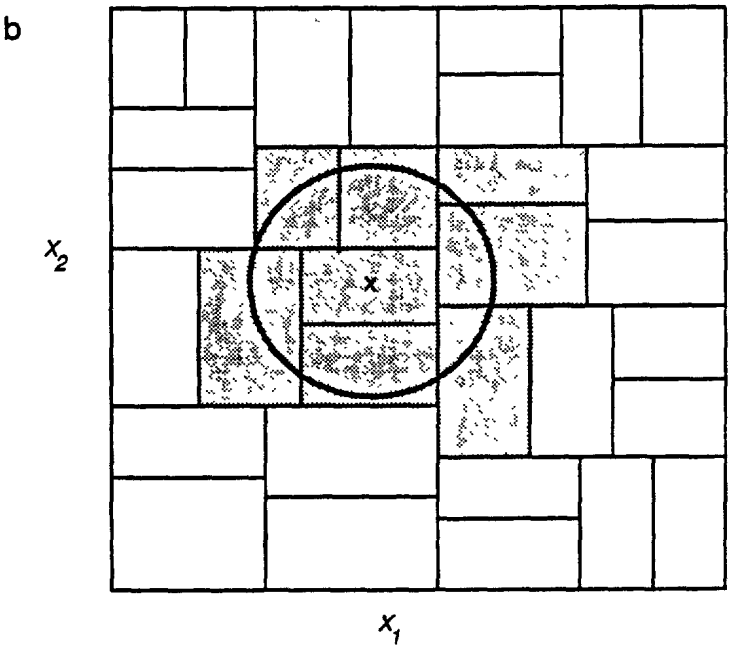


n = # patterns/bucket

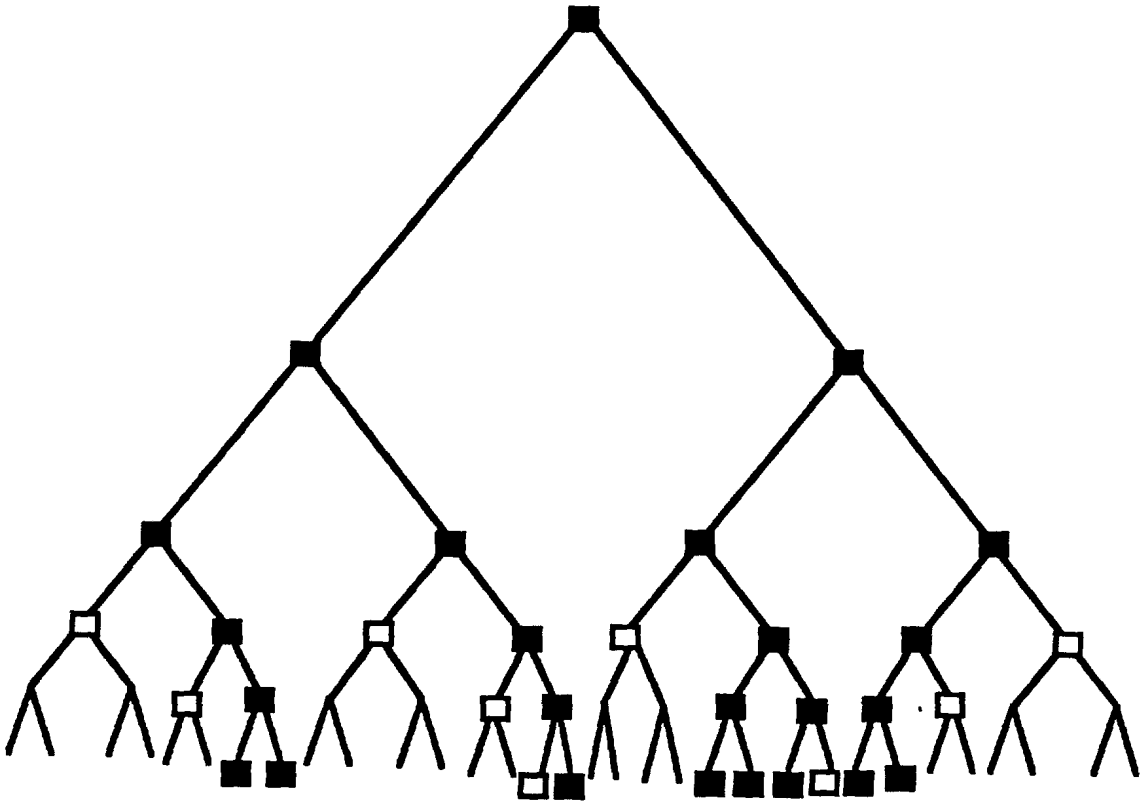
d = # search space dimensions

The ensemble of buckets the touch the search sphere for: (a) product partitioning, (b) k-d tree

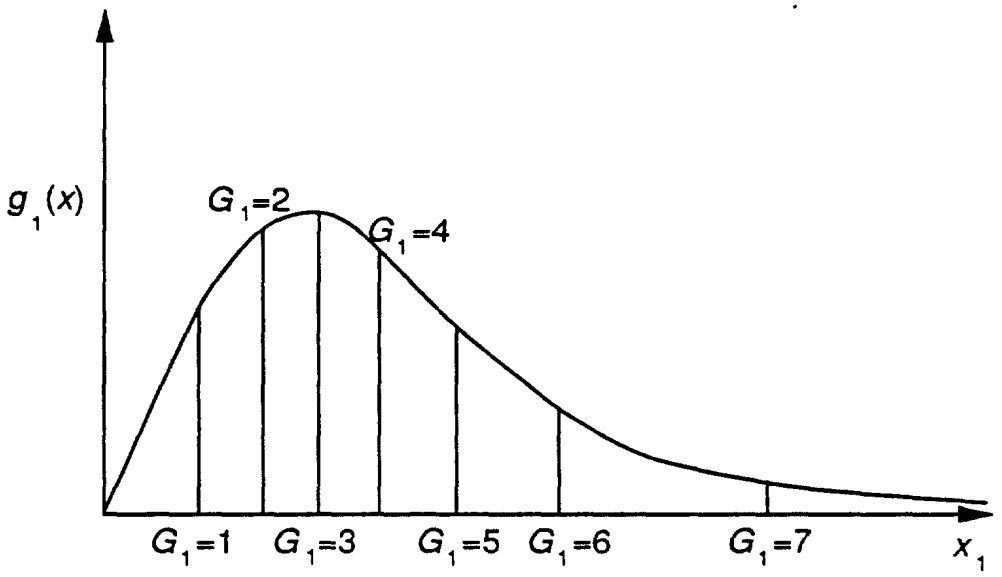




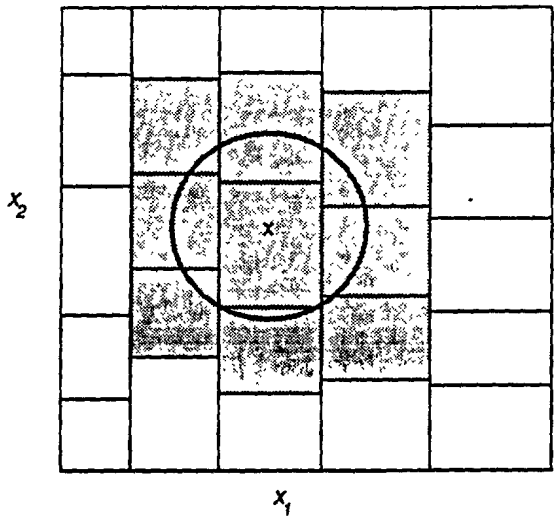
Binary search tree for k-d tree



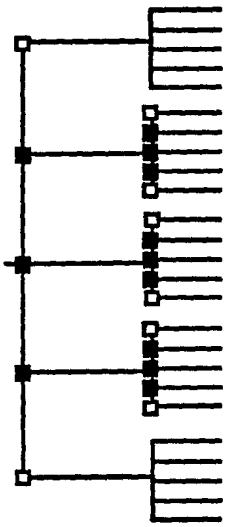
Pattern density function for placement of partitions in product partitioning



Ordered partitioning



Search tree for ordered partitioning



Ω_d = probability for a p-sphere of radius r of hitting a given bucket in a2-dimensional space

$$\Omega_2 = w_1 w_2 + 2r(w_1 + w_2) + \pi r^2$$

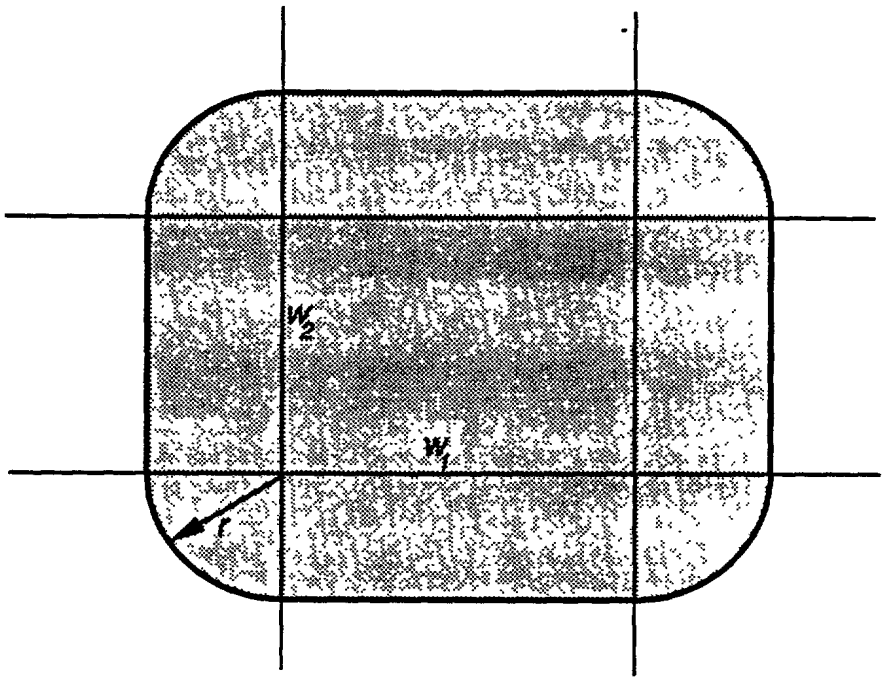
total number of buckets partitioning the p-sphere is

$$B_2 = \frac{\Omega_2}{w_1 w_2} = 1 + 2r\frac{(w_1 + w_2)}{w_1 w_2} + \frac{\pi r^2}{w_1 w_2}.$$

In d-dimensional space

$$B_d = \sum_{i=0}^{d} \frac{K_{p,i} r^i}{w^i} \binom{d}{i}$$

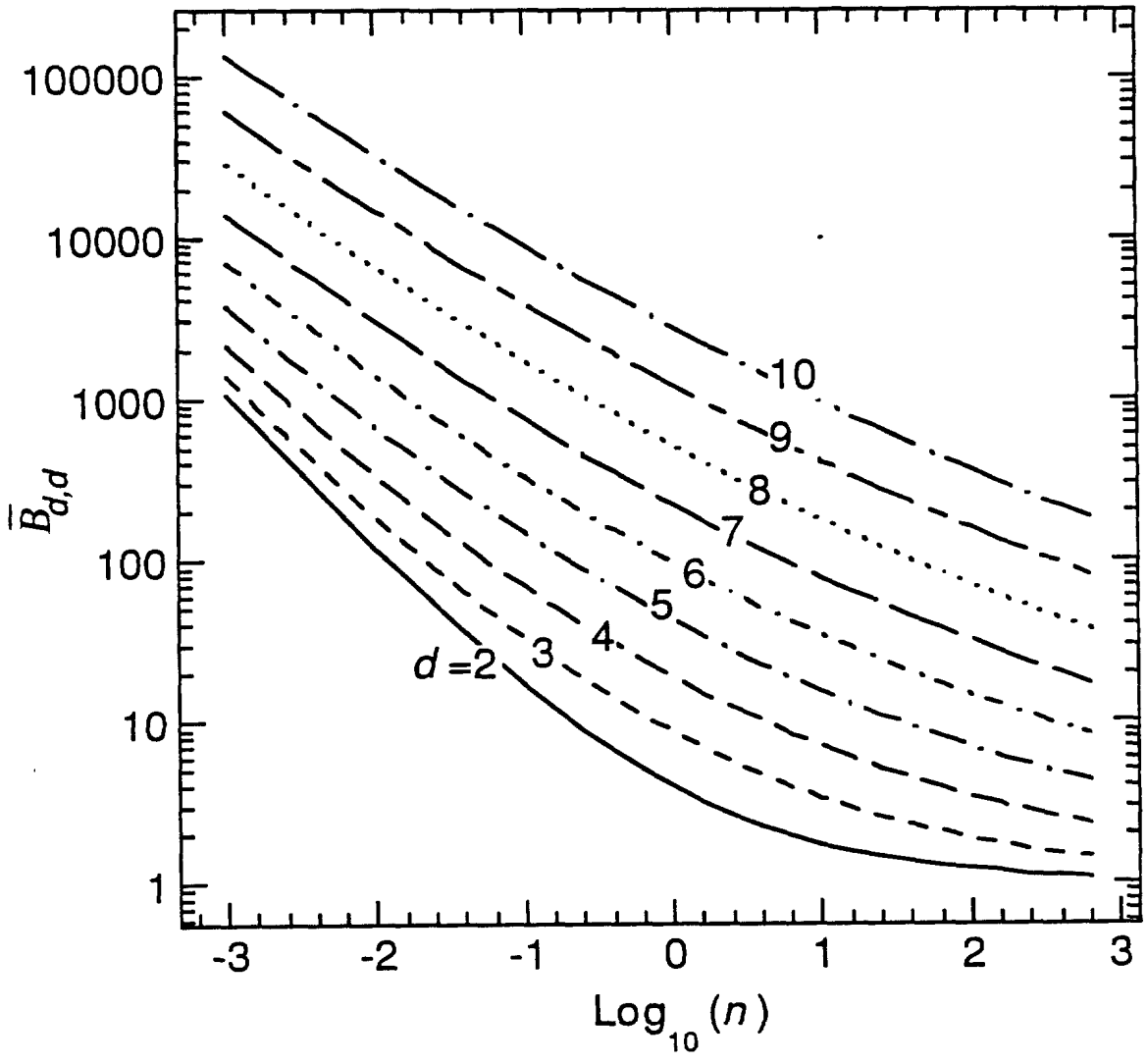
Region within a distance r of a bucket

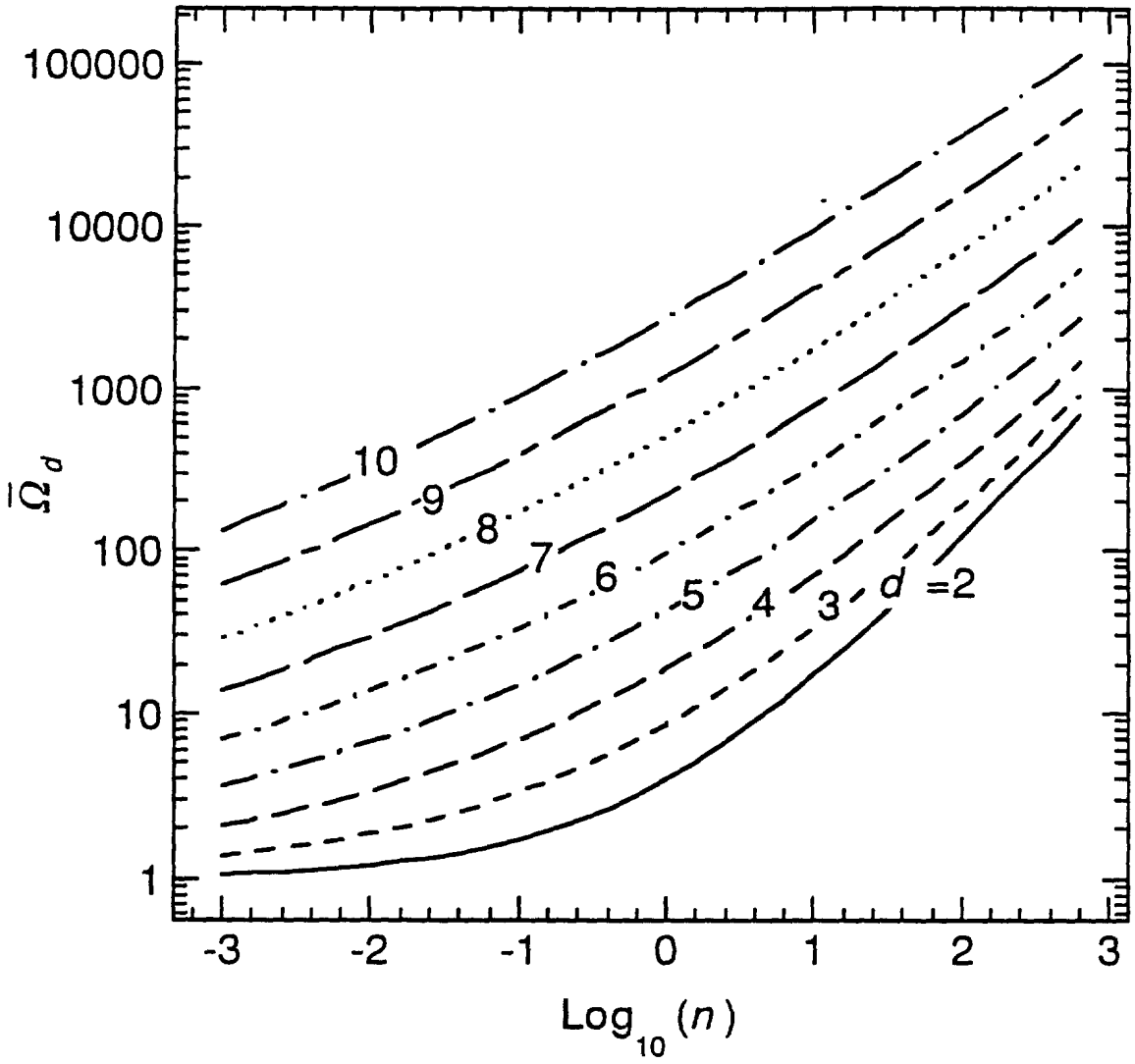


Expected number of j-dimensional buckets to search and patterns to compare

$$\overline{B}_{j,d} = E\left[B_j\right] = \sum_{i=0}^{j} \frac{\left(nK_{p,d}\right)^{-i/d} K_{p,i}}{(k-1)!} \binom{j}{i} \Gamma\left(\frac{i+dk}{d}\right)$$

$$\overline{\Omega}_d = n\overline{B}_{j,d}$$




100000
10000
1000
100
10
1
$\bar{\Omega}_d$
10
9
8
7
6
5
4
3
d =2
-3
-2
-1
0
1
2
3
$\mathrm{Log}_{10}(n)$

Total cost C_t:

$$C_t = C_o + \overline{\Omega}_d C_d$$

Product partitioning

$$C_o = \overline{B}_{d\text{-}1,d} C_r + \overline{B}_d C_b$$

Cylindric ordered partitioning

$$C_o = \sum_{i=1}^{d-1} \left(\overline{B}_{i,d} + 2\overline{B}_{i-1,d} \right) C_{d,i} + C_b \overline{B}_{d-1,d}.$$

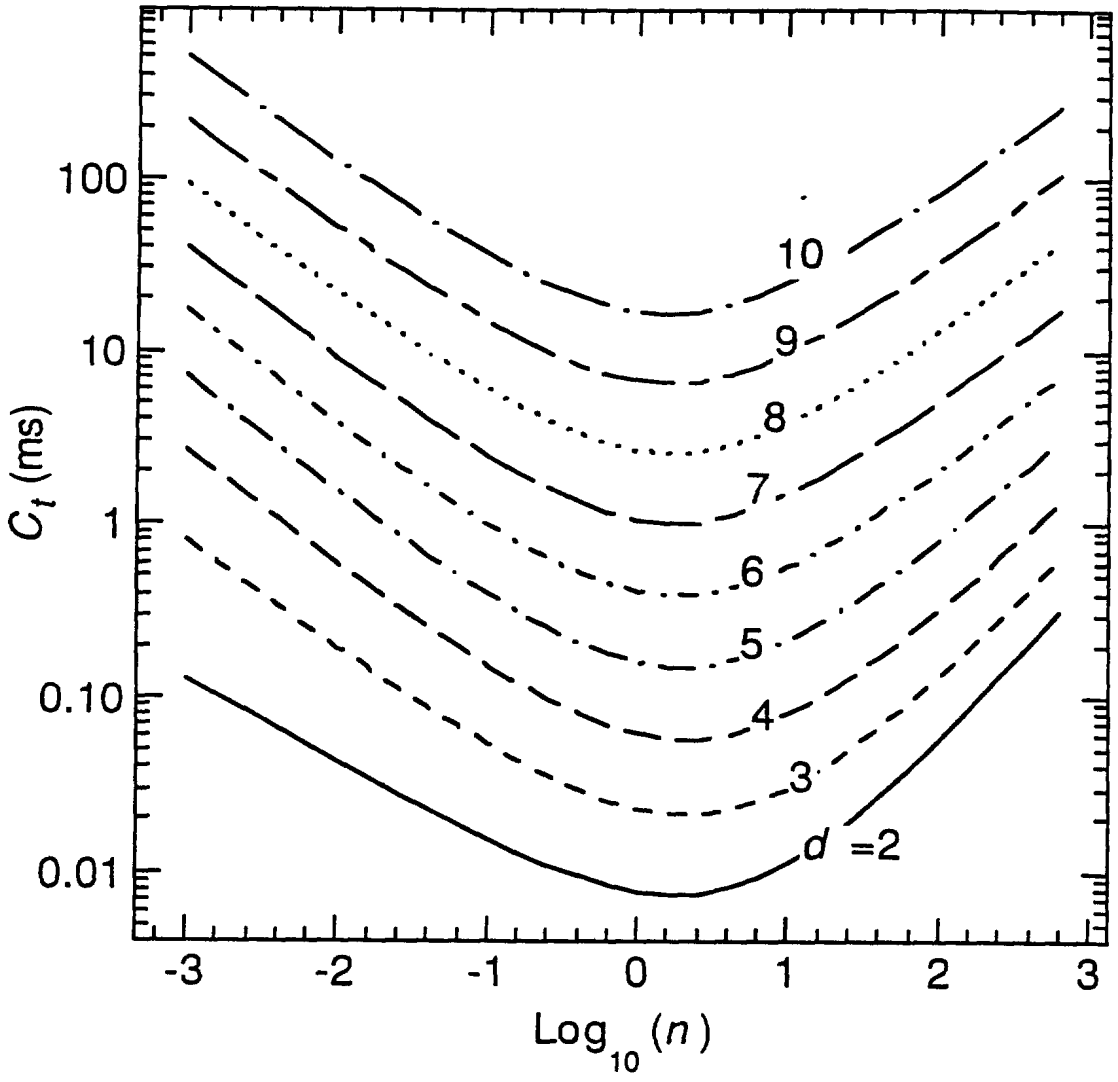
Limited ordered partitioning

$$C_o = \sum_{i=1}^{d-1} \left(\overline{B}_{i,d} + 2\overline{B}_{i-1,d} \right) C_{d,i} + C_b \overline{B}_{d-1,d} + C_r \overline{B}_{d-1,d}$$

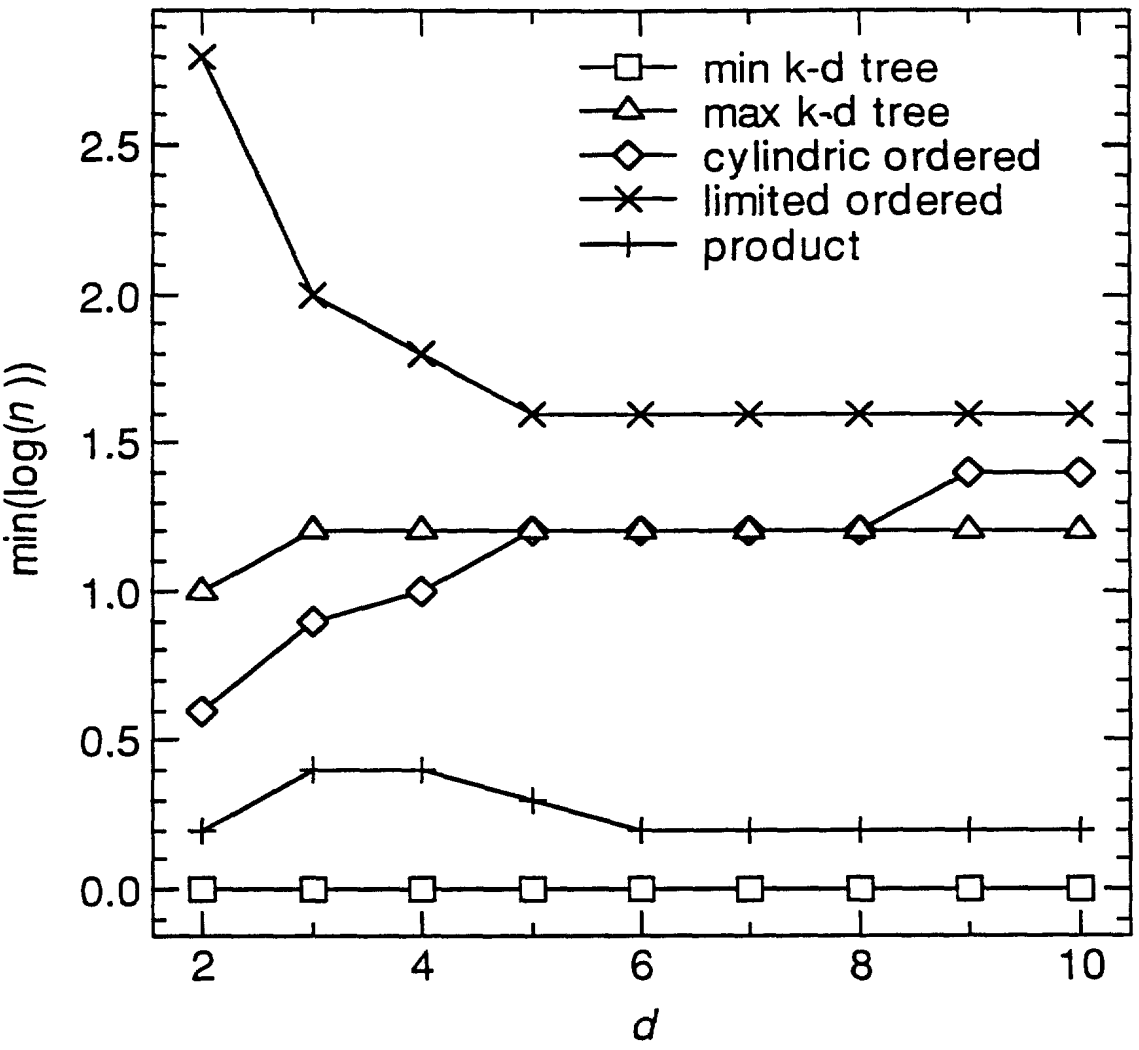
k-d tree

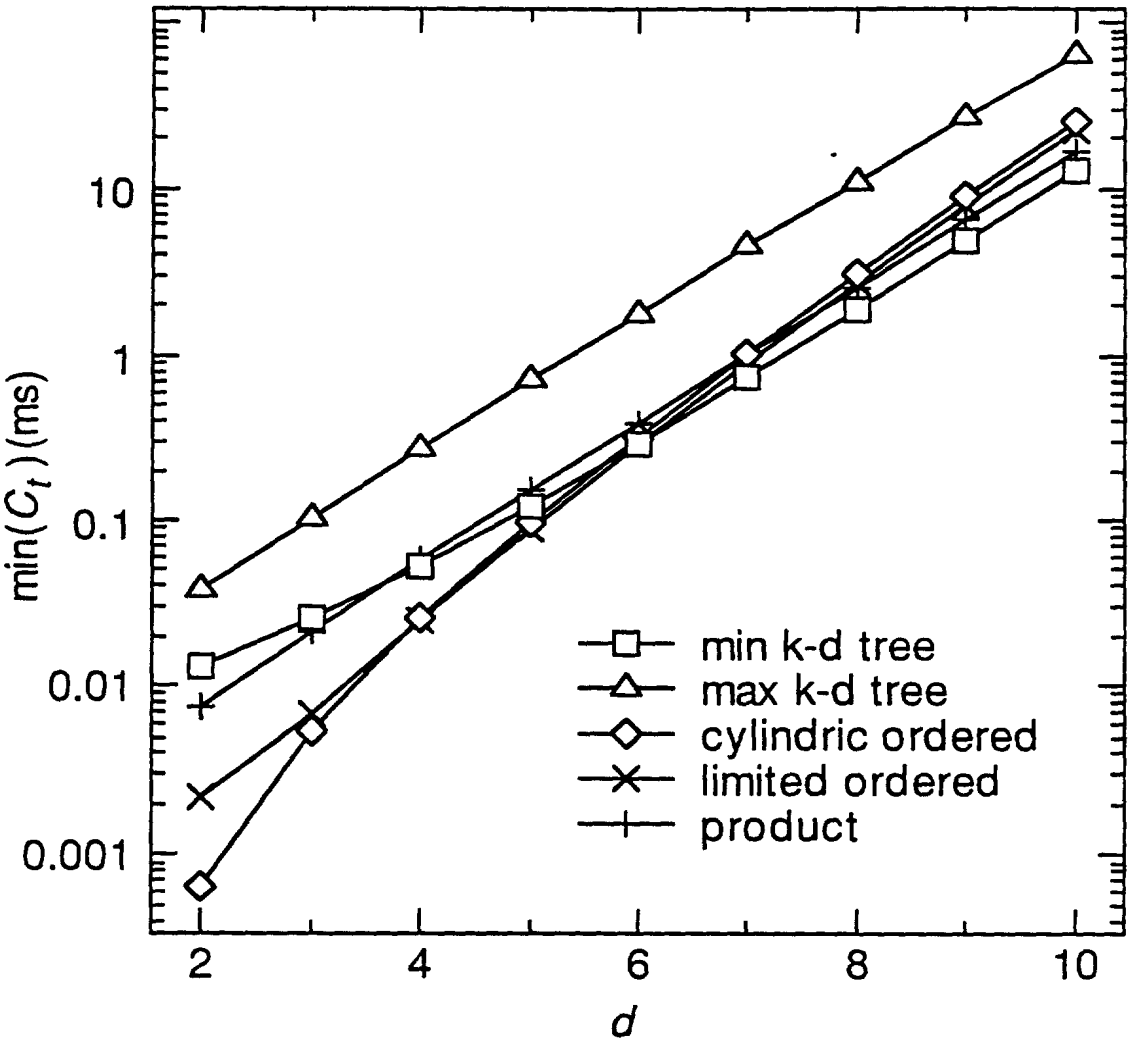
$$\min(N_i) = \min\left(\frac{\overline{B}_{d,d}}{2^{M-i+1}} + 1,\ 2^{i-1} \right)$$

$$\max(N_i) = \min\left(\overline{B}_{d,d},\ 2^{i-1} \right) \qquad M = \text{int}(\log_2 N) + 1$$



product partitioning


min k-d tree
max k-d tree
cylindric ordered
limited ordered
product
min(log(n))
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
8
10
d


10
1
0.1
0.01
0.001
min(C_t) (ms)
2
4
6
8
10
d
min k-d tree
max k-d tree
cylindric ordered
limited ordered
product

CONCLUSIONS

- Nearest-neighbour rule states that kNN is near the optimum classification rule, especially in high noise.

- Fast nearest-neighbour search divides search space into buckets, and pre-determine which pattern falls into which bucket. Equivalent to one-pass training, no generalization.

- Only those buckets close to the test pattern are searched.

- The optimum bucket size was calculated.

- Search time is asymptotically constant with number of patterns.

- In high d-space, all algorithms compared perform more or less the same. Not true for $d \leq 4$.