Exact L_∞ Nearest Neighbor Search in High Dimensions

Helmut Alt [*]
Institut für Informatik
Freie Universität Berlin
D-14195 Berlin, Germany
alt@inf.fu-berlin.de

Laura Heinrich-Litan [†]
Institut für Informatik
Freie Universität Berlin
D-14195 Berlin, Germany
litan@inf.fu-berlin.de

ABSTRACT

We present an algorithm for solving the nearest neighbor problem with respect to L_∞-distance. It requires no preprocessing and storage only for the point set P. Its average runtime assuming that the set P of n points is drawn randomly from the unit cube $[0,1]^d$ under uniform distribution is essentially $\Theta(nd/ln\ n)$ thereby improving the brute-force method by a factor of $\Theta(1/ln\ n)$. Several generalizations of the method are also presented, in particular to other "well-behaved" probability distributions and to the important problem of finding the k nearest neighbors to a query point.

1. INTRODUCTION

The *nearest-neighbor problem* is the problem of constructing an efficient data structure storing a set P of n points in $\mathbb{R}^d$ for answering *nearest neighbor queries*. In a nearest neighbor query some point $q \in \mathbb{R}^d$ is specified and the (some) point in P closest to q has to be determined.

The nearest neighbor problem apparently is a very natural geometric problem and it has numerous applications in e.g. statistics and data analysis [4] information retrieval [15], data compression [8], pattern recognition [7], and multimedia databases [14].

Various data structures for nearest neighbors have been developed in computational geometry, early ones by Dobkin and Lipton [6] and by Clarkson [3] based on higher-dimensional Voronoi-diagrams. However, although these data structures have query times logarithmic in n, it was assumed that the dimension d is a (low) constant and, in fact, the preprocessing time, the storage, or the query time are exponential in d. Only a structure by Meiser [13], more generally for point location in an arrangement of hyperplanes, has query time $O(d^5 \log n)$ but storage $O(n^{d+\varepsilon})$.

In many applications, however, dimension d of the search space is quite high and can reach several hundreds or even several thousands. Therefore, running times and storage requirements exponential in d are prohibitive in these cases. All query algorithms are in fact competing with the *brute-force* method of just determining the distance of q to each point in P and selecting the minimum. This method obviously requires no preprocessing, only the set P itself needs to be stored, and the query time is $O(nd)$ for all L_p-distances, $1 \le p \le \infty$.

In order to overcome the "curse of dimensionality" researchers looked for *c-approximate* solutions for the nearest neighbor problem, i.e. structures for efficiently finding points that are possibly not the nearest neighbor to the query point, but whose distance differs by at most a factor c from the minimal one. One of the early approximate solutions was given by Arya and Mount [1] which had still an exponential dependence on the dimension where the base depended on the quality of the approximation. The problem was investigated intensely in the last years and a real break-through was achieved by working with *randomized* techniques [11], [10], [12].

Finally,Indyk [9] presents a very sophisticated randomized data structure for the approximate nearest neighbor problem where preprocessing time, storage, and query time have no exponential dependence on the dimension any more.

In this paper we present an algorithm for solving the nearest-neighbor problem with respect to the L_∞-distance. It has the advantage that it solves the problem *exactly* and, furthermore, is very simple. Therefore, it is easy to implement and has a small constant in the O-term of its asymptotic runtime. Like the brute-force method it requires no preprocessing and also only the point set P itself is stored. Its average runtime assuming that the set P is drawn randomly from the unit cube $[0,1]^d$ under uniform distribution is essentially $\Theta(nd/ln\ n)$ thereby improving the brute-force method by a factor of $\Theta(1/ln\ n)$.

At first glance, this may not look like a significant improvement. However, the constant in the Θ-term is close to one, and for many applications n is in the range of tens or hundreds of thousands so the speed-up factor comes close to 10, which is a considerable improvement in practice. This consideration is confirmed by the experimental comparison of our method and the brute-force method in Section 5.

We will present several generalizations of our method. In particular to other "well-behaved" probability distributions

[*]Partially supported by Deutsche Forschungsgemeinschaft (DFG), grant AL 253/4-3.

[†]Supported by Deutsche Forschungsgemeinschaft (DFG), Graduiertenkolleg "Algorithmische Diskrete Mathematik", grant GRK 219/3.

SCG'01, June 3-5, 2001, Medford, Massachusetts, USA.

and to the important problem of finding the k nearest neighbors to a query point.

2. THE ALGORITHM

Throughout this paper we will assume that the set $P = \{p^1, \ldots, p^n\}$ is a fixed set of n points $p^i = (p_1^i, \ldots, p_d^i) \in \mathbb{R}^d$. Since we can translate and scale any given set P without changing the problem we will assume without loss of generality that $P \subset [0,1]^d$. For our probabilistic considerations we assume first that P is drawn at random from $[0,1]^d$ under uniform distribution. A generalization to other "well-behaved" distributions is presented in Section 4.

The idea of the our algorithms is the following. For a query $q = (q_1, \ldots, q_d) \in [0,1]^d$ consider the cube

$$C_{q,\alpha} = \left[q_1 - \frac{\alpha}{2}, q_1 + \frac{\alpha}{2}\right] \times \cdots \times \left[q_d - \frac{\alpha}{2}, q_d + \frac{\alpha}{2}\right]$$

around q of side length α such that the expected number of points in $C_{q,\alpha}$ is a low number φ. Suitable values for α and φ will be determined later. We scan the cube for the points contained in it: for each point p^i the algorithm scans its coordinates p_j^i. As soon as a point $p^i \in P$ turns out not to lie in C_α^d, the algorithm eliminates it from further consideration. The following gives a schematic description of the procedure for scanning the cube $C_{q,\alpha}$. The set P_α containing the points of the cube is returned as result.

SCAN_CUBE$(\alpha, \mathbf{q}, \mathbf{P})$

$P_\alpha = \emptyset$;
for $i = 1$ to n do
  $j = 1$;
  while ($j \le d$ and $|p_j^i - q_j| \le \frac{\alpha}{2}$) (TESTS)
  do $j = j + 1$;
  if ($j == d + 1$) then $P_\alpha = P_\alpha \cup \{p^i\}$;
return P_α ;

If $P_\alpha \neq \emptyset$ we determine the nearest neighbor of q by the brute-force method for the points of P_α. There is a nonzero probability that the cube $C_{q,\alpha}$ contains no points of P at all. In this case, the *brute-force* method will be called for all points of P. The parameter φ should be determined such that the probability of this event is so small that it does not effect the total asymptotic runtime. This search algorithm will be called the *cube-method*. The following gives a schematic description of this method.

CUBE_METHOD$(\mathbf{q}, \mathbf{P})$

α = DETERMINE_LENGTH(φ, q);
P_α = SCAN_CUBE(α, q, P);
if ($P_\alpha \neq \emptyset$)
  $\hat{p}$ = BRUTE_FORCE(P_α);
else
  $\hat{p}$ = BRUTE_FORCE(P); (BRUTE)

Procedure DETERMINE_LENGTH determines the side length α of a cube $C_{q,\alpha}$ with center q such that the expected number of points in $P \cap C_{q,\alpha}$ is φ. Details will be given later.

The SCAN_CUBE procedure can be improved as follows : keep decreasing α, whenever some point p^m lying closer to q is found. More precisely, if p^m turns out to be in the actual cube $C_{q,\alpha}$ then the actual nearest neighbor $\hat{p}$ is set to be p^m and the new side length α is set to be $2 \cdot \|p^m - q\|_\infty$. After this ADAPTIVE SCAN_CUBE procedure has scanned all points, $\hat{p}$ is the nearest neighbor of q if the scanned cube is not empty. We have some evidence that this variant improves the expected runtime of SCAN_CUBE by a constant factor but does not change its asymptotic complexity, see also our experimental comparison in Section 5. The analysis of the combined procedure is quite involved and currently under investigation.

3. EXPECTED RUNTIME ANALYSIS

For the analysis of the expected runtime of the algorithm we investigate the expected number of comparisons of the algorithm.

Let us first compute the expected number of comparisons with respect to the SCAN_CUBE procedure. For this we look at the expected number of tests performed in step (TESTS) for scanning a cube $C_{q,\alpha}$.

Let $Y_i = Y_i(\alpha)$ be the discrete random variable for the number of tests performed in step labeled by (TESTS) of SCAN_CUBE(α, q, P) for the point p^i and let $Y = Y(\alpha)$ be the random variable $Y = \sum_{i=1}^n Y_i$. The expected number of tests performed in step (TESTS) is given by $E[Y] = \sum_{i=1}^n E[Y_i]$.

Let $C_{q,\alpha}^j = \left[q_1 - \frac{\alpha}{2}, q_1 + \frac{\alpha}{2}\right] \times \cdots \times \left[q_j - \frac{\alpha}{2}, q_j + \frac{\alpha}{2}\right] \times [0,1]^{d-j}$. For the expected value $E[Y_i]$ of the number of tests with respect to the point $p^i \in P$ the following holds :

$$\begin{aligned} E[Y_i] &= \sum_{j=1}^{d} j \cdot \mathbf{Pr}[Y_i = j] = \sum_{j=1}^{d} \mathbf{Pr}[Y_i \ge j] \\ &= 1 + \sum_{j=1}^{d-1} \mathbf{Pr}\left[p^i \in C_{q,\alpha}^j\right], \end{aligned} \tag{1}$$

where $\mathbf{Pr}[Y_i \ge j]$ is the probability that the while-loop in (TESTS) is carried out at least j times ($1 \le j \le d$). Clearly, $\mathbf{Pr}[Y_i \ge 1] = 1$ and $\mathbf{Pr}[Y_i \ge j] = \mathbf{Pr}[p^i \in C_{q,\alpha}^{j-1}]$ ($1 < j \le d$). Because of uniform distribution $\mathbf{Pr}[p^i \in C_{q,\alpha}^j]$ equals the volume $\boldsymbol{V}(C_{q,\alpha}^j \cap [0,1]^d)$ of the box $C_{q,\alpha}^j \cap [0,1]^d$. We consider the simple case first:

Case $\mathbf{C_{q,\alpha} \subseteq [0,1]^d}$: $\mathbf{Pr}[p^i \in C_{q,\alpha}^j] = \boldsymbol{V}(C_{q,\alpha}^j) = \alpha^j$ and equation (1) implies for each $i = \{1, \ldots, n\}$:

$$E[Y_i] = 1 + \alpha + \alpha^2 + \ldots + \alpha^{d-1} \le \frac{1}{1-\alpha}. \tag{2}$$

The expected number φ of points in $C_{q,\alpha}$ is the product of $\boldsymbol{V}(C_{q,\alpha})$ with n, thus the side length

$$\alpha = \left(\frac{\varphi}{n}\right)^{\frac{1}{d}}$$

should be chosen in this case.

General case: Obviously, the volume of the box $W_{q,\alpha} = C_{q,\alpha} \cap [0,1]^d$ is the product of its side lengths. Let s_j be the side length of $W_{q,\alpha}$ in coordinate direction j:

$$s_j(\alpha) = \min(q_j + \alpha/2, 1) - \max(q_j - \alpha/2, 0)$$

which can be written as :

$$s_j(\alpha) = \begin{cases} \alpha & \text{if } 0 \le \alpha \le 2m(q_j) \\ m(q_j) + \frac{\alpha}{2} & \text{if } 2m(q_j) < \alpha \le 2 - 2m(q_j) \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

where $m(q_j) = \min(q_j, 1 - q_j)$.

So the volume of $W_{q,\alpha}$ is $\boldsymbol{V}(W_{q,\alpha}) = \prod_{j=1}^d s_j(\alpha)$. We will not compute α *exactly* so that $\boldsymbol{V}(W_{q,\alpha}) = \varphi/n$, since

that would involve determining the root of a polynomial of degree d, which is difficult and time consuming. Instead, since $\boldsymbol{V}(W_{q,\alpha})$ is monotone increasing in α the procedure DETERMINE_LENGTH computes some value α with

$$\frac{\varphi}{n} \leq \boldsymbol{V}(W_{q,\alpha}) = \prod_{j=1}^{d} s_j(\alpha) < \frac{\varphi+1}{n} \leq 1 \qquad (4)$$

by binary search. A detailed analysis shows that α can be determined by searching the interval $[\sqrt[d]{\varphi/n}, 2\sqrt[d]{\varphi/n}\,]$ in time $O(d\log(d\varphi))$. Thus, the DETERMINE_LENGTH procedure determines the side length α of the cube $C_{q,\alpha}$ such that the expected number of points in $P \cap C_{q,\alpha}$ is $\varphi' \in [\varphi, \varphi+1)$ in time $O(d\log(d\varphi))$.

For the expected number $E[Y_i]$ of tests we get :

$$E[Y_i] = 1 + s_1(\alpha) + \ldots + \prod_{l=1}^{j} s_l(\alpha) + \ldots + \prod_{l=1}^{d-1} s_l(\alpha)$$

Observe that some of the side lengths of $W_{q,\alpha}$ may be close to 1. To reduce the expected number of tests it is better to check those coordinates first where the box $W_{q,\alpha}$ has a small side length. Let $dim : \{1,\ldots,d\} \to \{1,\ldots,d\}$ be the bijection describing the order in which the coordinates will be checked in step (TESTS). We get

$$E[Y_i] = 1 + s_{dim(1)}(\alpha) + \cdots + \prod_{l=1}^{d-1} s_{dim(l)}(\alpha). \qquad (5)$$

We define dim such that

$$s_{dim(1)}(\alpha) \leq s_{dim(2)}(\alpha) \leq \cdots \leq s_{dim(d)}(\alpha).$$

Note that in this case the formula in (5) is minimal over all possible settings of dim. By formula (3) the following holds for the side lengths $s_j(\alpha)$:

$$s_j(\alpha) \leq s_l(\alpha) \iff \min(q_j, 1-q_j) \leq \min(q_l, 1-q_l)$$

Thus, the increasing order of the side lengths depends only on the query point q. The corresponding order dim can be computed once at the beginning in $O(d \log d)$ time.

In order to find a bound for $E[Y_i]$ we have to bound the partial products $\prod_{l=1}^{j} s_{dim(l)}(\alpha)$. Let λ be the geometric mean of the side lengths $s_j(\alpha)$ $j = 1,\ldots,d$. We claim that

$$\prod_{l=1}^{j} s_{dim(l)}(\alpha) \leq \lambda^j. \qquad (6)$$

Consider $\pi_k = \prod_{j=1}^{k} \beta_j$, where $\beta_j = \frac{s_{dim(l)}(\alpha)}{\lambda}$ for $j = 1,\ldots,d$. We get for some $l \leq d$ that

$$0 \leq \beta_1 \leq \ldots \beta_l \leq 1 \leq \beta_{l+1} \leq \cdots \leq \beta_d$$

consequently, $1 \geq \pi_1 \geq \cdots \geq \pi_l \leq \pi_{l+1} \leq \cdots \leq \pi_d = 1$ and therefore $\pi_k \leq 1$ for all $1 \leq k \leq d$ which is equivalent with inequality (6).

Because of $\lambda = \sqrt[d]{\boldsymbol{V}(W_{q,\alpha})} < 1$ we get by (6)

$$E[Y_i] \leq 1 + \lambda + \lambda^2 + \cdots + \lambda^{d-1} \leq \frac{1}{1-\lambda} \qquad (7)$$

Note that $\lambda = \alpha$ if special case $C_{q,\alpha} \subseteq [0,1]^d$ holds (see (2)).

Now we estimate the value of $\frac{1}{1-\lambda}$ where $\lambda = \sqrt[d]{\boldsymbol{V}(W_{q,\alpha})}$.

LEMMA 1. *Let $\lambda, L, U \in (0,1)$ with $L^{1/d} \leq \lambda \leq U^{1/d}$. Then the following inequalities hold :*

$$\frac{d}{\ln\left(\frac{1}{L}\right)} \leq \frac{1}{1-\lambda} \leq 2 \cdot max\left\{1, \frac{d}{\ln\left(\frac{1}{U}\right)}\right\}.$$

PROOF. Let $\beta = 1-\lambda$. Then $\frac{\ln(L)}{d} \leq \ln(1-\beta) \leq \frac{\ln(U)}{d}$ which implies

$$\frac{d}{\ln\left(\frac{1}{L}\right)} \leq \frac{1}{-\ln(1-\beta)} \leq \frac{d}{\ln\left(\frac{1}{U}\right)}.$$

By Taylor-expansion we have $\ln(1-\beta) = -\beta - \frac{\beta^2}{2}\frac{1}{\xi^2}$ for some $\xi \in [1-\beta, 1]$. Obviously, $\ln(1-\beta) < -\beta$ which implies $\frac{1}{-\ln(1-\beta)} < \frac{1}{\beta}$. Thus $\frac{d}{\ln\left(\frac{1}{L}\right)} < \frac{1}{1-\lambda}$.

In order to prove the second inequality of the lemma we consider the following cases :

a) Case $\lambda \leq 1/2$:

This implies $\frac{1}{1-\lambda} \leq 2 \leq 2 \cdot \max\left\{1, \frac{d}{\ln(1/U)}\right\}$.

b) Case $\lambda > 1/2$:

This implies $\beta < 1/2$, thus $\xi \geq 1-\beta > 1/2$ which implies $-\frac{\beta^2}{2}\frac{1}{\xi^2} > -2\beta^2 > -\beta$ and with this

$$\ln(1-\beta) = -\beta - \frac{\beta^2}{2}\frac{1}{\xi^2} > -2\beta.$$

Therefore $\frac{1}{1-\lambda} < 2\frac{d}{\ln(1/U)} \leq 2 \cdot \max\left\{1, \frac{d}{\ln(1/U)}\right\}$.

□

Inequations (4) and (7) and Lemma 1 imply that the expected number of tests $E[Y]$ required to scan the cube can be bounded as follows:

$$E[Y] \leq \max\left\{2n, \frac{2nd}{\ln(n/\varphi')}\right\} \leq \max\left\{2n, \frac{2nd}{\ln(n/(\varphi+1))}\right\}. \qquad (8)$$

There is a nonzero probability that the cube $C_{q,\alpha}$ contains no points of P at all. In this case, the *cube-method* calls the brute-force method which has a runtime of $\Theta(nd)$. We will now determine the parameter φ such that the probability of this event is so small that it does not effect the total asymptotic runtime.

The probability that no point of P is in $C_{q,\alpha}$ is $(1-\frac{\varphi'}{n})^n$ with $\varphi' \in [\varphi, \varphi+1)$. With probability $1-(1-\frac{\varphi'}{n})^n$ there is at least a point in $C_{q,\alpha}$ and in this case brute-force method will be called with the set P_α of the points in $C_{q,\alpha}$ having the expected runtime φd. Thus, the expected number T_{cube} of comparisons of the *cube-method* is given by :

$$\begin{aligned} T_{cube} &= E[Y] + (1-\frac{\varphi'}{n})^n nd + \left(1-(1-\frac{\varphi'}{n})^n\right)\cdot \varphi d \\ &\leq \max\left\{2n, \frac{2nd}{\ln(n/(\varphi+1))}\right\} + e^{-\varphi} nd + \varphi d \end{aligned}$$

In order that the expected asymptotic runtime of the brute-force part of the algorithm does not become worse than the

runtime required to scan the cube it is sufficient to choose φ such that

$$e^{-\varphi} \cdot nd \leq \frac{nd}{ln(n/(\varphi+1))} \tag{9}$$

(9) is equivalent to $\varphi \geq ln\ ln(n/(\varphi+1))$ which is satisfied, if we choose

$$\varphi \geq ln\ ln\ n.$$

Summarizing, we get

THEOREM 1. *Let P be a set of n points from $[0,1]^d$. The cube method finds the nearest neighbor from P to some query point $q \in [0,1]^d$ with an expected asymptotic runtime of $O\left(\frac{nd}{\ln n} + n + d\log(d \ln\ln n)\right)$ if the points of P are independently drawn at random from $[0,1]^d$ under uniform distribution.*

4. EXTENSIONS

k-Nearest-Neighbor Search

If the scanned cube $C_{q,\alpha}$ is empty the alternative to determining the nearest neighbor by brute-force is to consider a bigger cube with center q and scan it for points. The size of the current cube should be increased as long as it contains no points of P. This search algorithm will be called the *growing-cube method.*

The *growing-cube method* can be extended to compute the k nearest neighbors of the query point q from the point set P. The extended *growing-cube method* for k nearest neighbor search works as follows. The first cube C_{q,α_1} around q which contains an expected number of about k points is scanned for points. If the cube contains less than k points then a bigger cube is considered. φ_t is the expected number of points which are contained in the cube considered in the t-th iteration. The side length α_t of this cube C_{q,α_t} is determined as in Section 3 by the procedure DETERMINE_LENGTH. How to choose an appropriate sequence $k \leq \varphi_1 < \varphi_2 < \ldots < \varphi_t < \ldots$ will be shown later. The following gives a schematic description of the algorithm.

GROWING_CUBE ($\mathbf{k}, \mathbf{q}, \mathbf{P}$)
$t = 0$; $L_0 = \emptyset$
repeat
  t=t+1;
  if ($\varphi_t < n$) then
    α_t = DETERMINE_LENGTH(φ_t, q);
    P_t = SCAN_CUBE(α_t, q, P);
    $L_t = L_{t-1} \cup P_t$; $P = P \setminus P_t$;
  else $L_t = P$; $P_t = P \setminus L_{t-1}$;
until ($|L_t| \geq k$);
if ($|L_t| > k$)
  then $L = L_{t-1} \cup$ SELECT($k - |L_{t-1}|, q, P_t$); (SELECT)
else $L = L_t$;

The set L contains the k nearest neighbors to be found. If there were more then k points in L_t then in the t-th iteration there were more points found as needed. We extract the extra $k - |L_{t-1}|$ nearest neighbors that we actually need from the set P_t by the SELECT-procedure. The procedure SELECT(k', q, P') computes the distances of all points of P' to the query q, selects the k' smallest distances and returns as result the corresponding points from P'. Using a linear time selection algorithm the running time of SELECT(k', q, P') is $c \cdot d \cdot |P'|$ for some appropriate constant $c > 0$.

Let Y^t be the discrete random variable for the number of tests performed to scan the cube C_{q,α_t}. And let X^t be the discrete random variable for the number $|L_t|$ of points found in the first t iterations. Let T_{scan} be the expected number of tests required to scan the cubes. In a straightforward manner it can be shown that $E[Y^t | X^{t-1} < k] \leq E[Y^t]$, thus T_{scan} is bounded by:

$$\begin{aligned} T_{scan} &\leq \sum_{t\geq 1} \mathbf{Pr}(X^{t-1} < k) \cdot E[Y^t \mid X^{t-1} < k] \\ &\leq \sum_{t\geq 1} \mathbf{Pr}(X^{t-1} < k) \cdot E[Y^t] \end{aligned} \tag{10}$$

The expected asymptotic runtime T_{sel} of step (SELECT) can be bounded as follows:

$$\begin{aligned} T_{sel} \leq \sum_{t\geq 1} &\mathbf{Pr}(X^{t-1} < k) \cdot \mathbf{Pr}(X^t > k \mid X^{t-1} < k) \\ &\cdot c \cdot d \cdot E[X^t - X^{t-1} | X^{t-1} < k, X^t > k] \end{aligned}$$

where c is the constant of the SELECT-procedure. It can be proven that

$$\begin{aligned} T_{sel} &\leq c \cdot d \cdot \sum_{t\geq 1} \mathbf{Pr}(X^{t-1} < k+1) \cdot E[X^t] \\ &= c \cdot d \cdot \sum_{t\geq 1} \mathbf{Pr}(X^{t-1} < k+1) \cdot \varphi_t. \end{aligned} \tag{11}$$

Now we bound the probability $\mathbf{Pr}(X^t < k)$ using the following theorem (see [5]) which is based on the Chernoff-Bound technique.

THEOREM 2. [5] *Let Z be the discrete variable which counts the total number of successes by performing n independent Bernoulli trials with the success probability p of a trial. Then*

$$\mathbf{Pr}(Z - np \geq r) \leq e^{\lambda^2\sigma^2}/e^{\lambda r}$$

for each $\lambda < \frac{1}{\max(p,1-p)}$ where $\sigma^2 = np(1-p)$ is the variance of Z.

COROLLARY 1. *Taking $\lambda := \frac{r}{2n(1-p)}$ in Theorem 2 we get the bound:*

$$\mathbf{Pr}(Z - np \geq r) \leq e^{\frac{-r^2}{4n(1-p)}}$$

where $0 < r < \frac{2n(1-p)}{\max(p,1-p)}$.

Let N^t be the random variable for the number of points of P within $[0,1]^d \setminus C^d_{\alpha_t}$. Thus, $N^t = n - X^t$, $\mathbf{Pr}(X^t < k) = \mathbf{Pr}(N^t > n-k) = \mathbf{Pr}(N^t \geq n-k+1)$ and for $p' \in [0,1]^d$ the probability $p = \mathbf{Pr}(p' \in N^t)$ equals $1 - \frac{\varphi_t}{n}$. Let

$$r := n - k + 1 - np = n - k + 1 - (n - \varphi_t) = \varphi_t - k + 1$$

Because of

$$0 < r = \varphi_t - k + 1 \leq \varphi_t = n(1-p) < \frac{2n(1-p)}{\max(p, 1-p)}$$

we get by Corollary 1 :

$$\begin{aligned}\mathbf{Pr}(N^t \geq n-k+1) &= \mathbf{Pr}(N^t - np \geq n-k+1-np) \\ &\leq e^{\frac{-r^2}{4n(1-p)}} = e^{\frac{-(\varphi_t-k+1)^2}{4\varphi_t}}.\end{aligned}$$

Because of $\frac{-(\varphi_t-k+1)^2}{4\varphi_t} \leq -\frac{\varphi_t}{4} + \frac{k-1}{2}$ we get

$$\mathbf{Pr}(X^t < k) \leq e^{-\varphi_t/4+(k-1)/2}. \tag{12}$$

Note that if φ_t is chosen to grow strictly monotone with t then the *repeat*-loop in the GROWING-CUBE procedure terminates. We choose $\varphi_t = 2(k-1)+t$. In this case by (12)

$$\mathbf{Pr}(X^t < k) \leq e^{-t/4}. \tag{13}$$

Now let us bound $E[X^t]$. Clearly, $E[Y^t] \leq nd$ and by (8) $E[Y^t] \leq \max\{2n, \frac{2nd}{\ln(n/\varphi_t)}\}$. If $\varphi_t \leq \sqrt{nk}$ then:

$$\frac{nd}{\ln(n/\varphi_t)} \leq \frac{nd}{\ln(n/\sqrt{nk})} = \frac{2nd}{\ln(n/k)}$$

Thus, using (10) T_{scan} can be bounded as follows :

$$\begin{aligned}T_{scan} \leq & \sum_{\varphi_t \leq \lfloor\sqrt{nk}\rfloor} \mathbf{Pr}(X^{t-1} < k) \cdot \max\{2n, \frac{4nd}{\ln(n/k)}\} \\ & + \sum_{\varphi_t > \lfloor\sqrt{nk}\rfloor} \mathbf{Pr}(X^{t-1} < k) \cdot nd\end{aligned}$$

If $\varphi_t \geq \lfloor\sqrt{nk}\rfloor$ we have $-\frac{t}{4} = -\frac{\varphi_t}{4} + \frac{k-1}{2} \leq -\frac{\lfloor\sqrt{nk}\rfloor}{4} + \frac{k-1}{2}$. It can be shown that $\frac{\lfloor\sqrt{nk}\rfloor}{4} - \frac{k-1}{2} \geq \frac{\sqrt{n}}{4} \geq \ln n$ for n big enough and $k \leq n/8$. In this case, if $\varphi_t \geq \lfloor\sqrt{nk}\rfloor$ then $-\frac{t}{4} \leq -\ln n$. Let t_* be the smallest t such that $\varphi_t \geq \lfloor\sqrt{nk}\rfloor$. Since $-\frac{t_*}{4} \leq -\ln n$, we have by (13):

$$\begin{aligned}\sum_{\varphi_t > \lfloor\sqrt{nk}\rfloor} \mathbf{Pr}(X^{t-1} < k) &= \sum_{\varphi_t \geq \lfloor\sqrt{nk}\rfloor} \mathbf{Pr}(X^t < k) \\ &\leq \sum_{t \geq t_*} e^{-t/4} = e^{-t_*/4} \cdot O(1) \\ &\leq \frac{1}{n} \cdot O(1)\end{aligned}$$

Because of (13) we get:

$$\begin{aligned}\sum_{\varphi_t \leq \lfloor\sqrt{nk}\rfloor} \mathbf{Pr}(X^{t-1} < k) &\leq \sum_{t \geq 0} \mathbf{Pr}(X^t < k) \\ &\leq \sum_{t \geq 0} e^{-t/4} = O(1)\end{aligned}$$

With this we get for $k \leq n/8$ $T_{scan} = O\left(\frac{nd}{\ln(n/k)} + n\right)$. Notice that our method is only of interest if k is significantly smaller then n, since $\Theta(kd)$ tests are needed in any case and the brute-force method cannot be beaten asymptotically if $k = \Theta(n)$.

The estimation of T_{sel} in (11) works analogously. We have by (12) $\mathbf{Pr}(X^t < k+1) \leq e^{-\varphi_t/4+k/2}$. For $\varphi_t = 2(k-1)+t$ we get $\mathbf{Pr}(X^t < k+1) \leq e^{-t/4+1/2}$, thus by (11):

$$\begin{aligned}T_{sel} &\leq c \cdot d \cdot \sum_{t \geq 0} \mathbf{Pr}(X^t < k+1) \cdot \varphi_{t+1} \\ &= c \cdot d \cdot \sum_{t \geq 0} \mathbf{Pr}(X^t < k+1) \cdot (2k-1+t) \\ &\leq c \cdot d \cdot \sum_{t \geq 0} e^{1/2} \cdot e^{-t/4} \cdot (2k-1+t)\end{aligned}$$

Because of $\sum_{t \geq 0} e^{-t/4} = O(1)$ and $\sum_{t \geq 0} t \cdot e^{-t/4} = O(1)$ we get $T_{sel} = O(dk)$.

Analogously, it can be shown that the asymptotic runtime T_{side} for the computation of the side lengths of the cubes to be scanned can be bounded by :

$$T_{side} \leq \sum_{t \geq 1} \mathbf{Pr}(X^{t-1} < k) \cdot c_S \cdot d \cdot \log(d\varphi_t) = O(d \log(kd))$$

where c_S is the constant of the DETERMINE_LENGTH procedure introduced in Section 3.

THEOREM 3. *Let P be a set of n points from $[0,1]^d$. The growing-cube method finds the k nearest neighbors from P to some query point $q \in [0,1]^d$ with an expected asymptotic runtime of $O\left(\frac{nd}{\ln(n/k)} + n + dk + d\log d\right)$ if the points of P are independently drawn at random from $[0,1]^d$ under uniform distribution.*

Other distributions

Let $K = (K_1, \ldots, K_d)$ be the random vector for the coordinates of a point $p \in P$ and let $F_K : \mathbb{R}^d \to [0,1]$ be the *joint distribution function* of the random vector K. We consider the points from P to be independently drawn at random under the following distributions:

a) Independent distribution:
The random variables $K_1, \ldots, K_d$ are independent. We have $F_K(x_1, \ldots, x_d) = \prod_{j=1}^d F_{K_j}(x_j)$, where F_{K_j} is the distribution function of the random variable K_j.

In this case we have :

$$\mathbf{Pr}[p^i \in C^j_{q,\alpha}] = \prod_{l=1}^{j} \left(F_{K_l}(q_l + \alpha/2) - F_{K_l}(q_l - \alpha/2) \right).$$

Let denote $s_l(\alpha) = F_{K_l}(q_l + \alpha/2) - F_{K_l}(q_l - \alpha/2)$ for $l = 1, \ldots, d$. Note that the functions $s_l : \mathbb{R}_+ \to [0,1]$ are continuous and monotone increasing in α. This implies that $\prod_{l=1}^d s_l(\alpha)$ is continuous and monotone increasing in α. Thus some value α such that

$$\frac{\varphi}{n} \leq \mathbf{Pr}\left[p^i \in Cq, \alpha\right] = \prod_{j=1}^{d} s_j(\alpha) < \frac{\varphi+1}{n} \leq 1$$

could be determined by binary search. For this value α the expected number of points in Cq, α is $\varphi' \in [\varphi, \varphi+1)$. If the bijection *dim* is choosen such that

$$s_{dim(1)}(\alpha) \leq \ldots \leq s_{dim(d)}(\alpha)$$

by Lemma 1 we get

$$E[Y] \leq \frac{n}{1-\lambda} \leq \max\left\{2n, \frac{2nd}{\ln(n/\varphi)}\right\}$$

where $\lambda = \sqrt[d]{\prod_{j=1}^d s_j(\alpha)}$. Analogously, as in Section 3 it can be shown that the expected number T_{cube} of coordinate

tests required by the the *cube-method* is $O(\frac{nd}{\ln}+n)$. Besides T_{cube} the total runtime *cube-method* includes the costs for computing the side length α, which depends on the complexity of computing the distribution functions F_{K_j}.

b) Positive-bounded distribution(see [2]):
We say that $K=(K_1,\ldots,K_d)$ is positive-bounded over the bounded, convex open region $\boldsymbol{G}$ if there exists constants $0<C_1\le C_2$ such that $C_1\le f_K(x)\le C_2$ for all $x\in\boldsymbol{G}\setminus A$ where A has Lebesgue measure $m(A)=0$ and $f_K(x)=0$ for $\mathbb{R}^d\setminus\boldsymbol{G}$. With other words, F_K has a density with respect to Lebesgue measure that is bounded above and bounded below away from zero. The distribution function has the following property:

$$C_1\cdot\boldsymbol{V}(S)\le\mathbf{Pr}[x\in S]\le C_2\cdot\boldsymbol{V}(S)$$

where $S\subseteq\boldsymbol{G}$ is a region of volume (measure) $\boldsymbol{V}(S)$.

As mentioned in Section 2 we can assume w.l.o.g. $P\subseteq[0,1]^d$. For $\boldsymbol{G}=(0,1)^d$ we get for each point $p^i\in P$

$$\mathbf{Pr}\left[p^i\in C^j_{q,\alpha}\right]=\mathbf{Pr}\left[p^i\in W^j_{q,\alpha}\right]$$

and therefore,

$$C_1\cdot\boldsymbol{V}(W^j_{q,\alpha})\le\mathbf{Pr}\left[p^i\in C^j_{q,\alpha}\right]\le C_2\cdot\boldsymbol{V}(W^j_{q,\alpha}).$$

where the box $W^j_{q,\alpha}$ is $C^j_{q,\alpha}\cap[0,1]^d$.Therefore,

$$C_1\cdot\prod_{l=1}^{j}s_l(\alpha)\le\mathbf{Pr}[p^i\in C^j_{q,\alpha}]\le C_2\cdot\prod_{l=1}^{j}s_l(\alpha)$$

where $s_l(\alpha)$ is the side length in coordinate direction l of $W_{q,\alpha}$ and is given by formula (3). For the side length α and the bijection *dim* choosen as in Section 3 we get by (1) and by Lemma 1:

$$\begin{aligned}E[Y] &\le \sum_{j=0}^{d-1}\mathbf{Pr}\left[p^i\in C^j_{q,\alpha}\right]\le C_2\cdot\sum_{j=0}^{d-1}\lambda^j\\ &\le \max\left\{2C_2n,\frac{2C_2nd}{\log(n/(\varphi+1))}\right\}\end{aligned}$$

where $\lambda=\sqrt[d]{\boldsymbol{V}(W_{q,\alpha})}$. For the probability of failure we have $\mathbf{Pr}(P\cap C_{q,\alpha}=\emptyset)\le(1-C_1\frac{\varphi}{n})^n\le e^{-C_1\varphi}$. For suitable value of $\varphi\ge\frac{1}{C_1}\cdot\ln(C_2\ln n)$ the expected asymptotic runtime of the *cube-method* is $O(\frac{nd}{\ln}+n+d\log(d\ln\ln n))$ if the points are drawn at random from $[0,1]^d$ under positive-bounded distribution.

5. EXPERIMENTS

For the experiments the data set P was generated in $[0,1]^d$ according to the uniform distribution.

Figure 1 shows an experimental comparison of the implementations of the variants of the algorithm for dimension $d=100$ and sizes of P between 1000 and 10000. *adaptcube* and *adaptgrow* are the variants of the cube-method and of the growing-cube method, respectively which use the improved ADAPTIVE SCAN_CUBE procedure (see Section 2). The value on the vertical scale is the factor by which each algorithm is faster then the brute-force method.

Figure 2 shows a comparison of the *growing-cube* method for k nearest neighbor search with the brute-force method, which computes all distances of the points of P to the query and selects the k smallest distances among them.

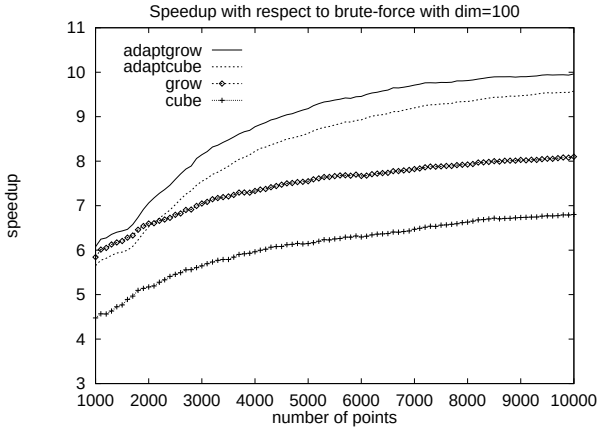

Figure 1: Comparison of the algorithms with the brute-force method

Figure 2: Comparison of the *growing-cube* method for k nearest neighbor search with the brute-force method

6. REFERENCES

[1] S. Arya and D. M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 271–280, 1993.

[2] J. L. Bentley, B. W. Weide and A. C. Yao. Optimal expected time algorithms for closest point problems In ACM Trans. on Math. Software 6, pages 563–580, 1980.

[3] K. L. Clarkson. A randomized algorithm for closest-point queries. *SIAM J. Comput.*, 17, pages 830–847, 1988.

[4] L. Devroye and T. Wagner. Nearest neighbor methods in discrimination. In North-Holland, editor, *Handbook of Statistics*, volume 2. P.R. Krishnaiah, L.N. Kanal, 1982.

[5] D. P. Dubhashi and A. Panconesi. Concentration of Measure for the Analysis of Randomized Algorithms. Manuscript, pages 13–14, http://www.dsi.uniroma1.it/~ale/papers.html.

[6] D. P. Dobkin and R. J. Lipton. Multidimensional searching problems. *SIAM J. Comput.*, 5, pages 181–186, 1976.

[7] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* Wiley, 1973.

[8] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Press, 1991.

[9] P. Indyk. Dimensionality reduction techniques for proximity problems. In *Proceedings, ACM Symposium on Data Structures and Algorithms, SODA 2000*, pages 371–378, 2000.

[10] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Annu. ACM Sympos. Theory Comput.*, pages 604–613, 1998.

[11] J. Kleinberg. Two algorithms for nearest-neighbor search in high dimension. In *Proc. 29th Annu. ACM Sympos. Theory Comput.*, pages 599–608, 1997.

[12] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. In *Proc. 30th Annu. ACM Sympos. Theory Comput.*, pages 614–623, 1998.

[13] S. Meiser. Point location in arrangements of hyperplanes. *Inform. Comput.*, 106, pages 286–303, 1993.

[14] A. W. M. Smeulders and R. Jain (eds.). Image databases and multi-media search. In *Proceedings of the First International Workshop, IDB-MMS '96*, Amsterdam, 1996. Amsterdam University Press.

[15] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, 1979.