# Nearest Neighbor Search in High Dimensional Space by Using Convex Hulls[1]

Ingo Schmitt

Otto-von-Guericke-Universität Magdeburg
Institut für Technische Informationssysteme
Postfach 4120, D–39016 Magdeburg, Germany
E-mail: schmitt@iti.cs.uni-magdeburg.de

February 28, 2001

[1]This research was supported by the Natural Science and Engineering Research Council (NSERC) of Canada and a University of Windsor grant.

**Abstract**

The problem of finding nearest neighbors in high dimensional space is important for a broad spectrum of applications. Many research and experimental results reveal the curse of dimensionality. In this paper, based on statistical results, we give a deep understanding of this phenomenon by relating retrieval costs to the development of approximation errors and the standard deviation of distances from a query point. In contrast to other approaches we use convex hulls which are minimal convex clusters for a given set of data points. Our way of understanding the curse helps to focus research on worthwhile topics and to estimate retrieval costs for different data distributions, cluster shapes and distance functions. We then discuss the approach of manipulating distance functions and use the power of our retrieval model to judge the efficiency of new distance functions.

# Contents

# Chapter 1

# Introduction

Nearest neighbor search in high-dimensional space is an important problem for a wide spectrum of applications. For example, it is important for similarity retrieval in multimedia databases [Lu99] where multimedia objects are mapped to points in high dimensional space and a distance function measures the dis-similarity among them. There is a strong need for efficient data structures which find nearest neighbors to a given query point faster than checking every point sequentially (linear cost). If the number of dimensions is very low then there exist data structures which work in nearly logarithmic time (see [GG98] for an overview). But all data structures fails if the number of dimensions is too high ($d > 20$) (cf. for example [WSB98]). This problem is often called the *curse of dimensionality*.

[BGRS99] discuss whether nearest neighbor search in high dimensional space is meaningful. Since the distance to the nearest neighbor approaches the distance to the farthest neighbor the distinction between them blurs with as few as 15 dimensions. In our paper we show a nearly constant difference between the farthest and the nearest $L_2$-distance as the number of dimensions grows. Following our model we show that besides this difference the ratio between the farthest and the nearest distance plays an important role for the retrieval cost.

[FBF77, BBKK97, BBK+01] try to find the reason for the curse of dimensionality by developing cost models. They show that the nearest neighbor distance grows as the number of dimensions increases and that the number of pages to be visited tends towards 100%. However, they assume box-shaped clusters and let the question open whether another cluster shape would be less unsuccessful.

In this paper we use a convex hull cluster which is a minimal convex cluster. Since every convex cluster shape (e.g. spheres, boxes, ellipsoids) contains the convex hull our retrieval cost can be regarded as an absolute lower bound. Furthermore, we base our retrieval costs on the concept of the approximation error and the difference between the maximal and minimal distance. Both concepts depend in turn on statistical concepts as the mean $\mu_d$ and the standard deviation $\sigma_d$ of distances between points in high dimensional space. Our discussion explains the reason of the curse by use of a very simple model and gives hints for further research directions. Very interesting are approaches which cause better statistical values for $\mu_d$ and $\sigma_d$. A straightforward approach is to analyze different distance functions. [AHK01] recommend to use a $L_k$ metric with a low value for $k$. We will discuss the impact of different distance functions on the retrieval costs at the end of this paper.

In the next section we develop the expecting values and the standard deviations for different distance functions in dependence on the number of dimensions. For our analysis we use uniformly distributed points in $[0..1]^d$. Section 3 is the core section where the retrieval costs are related to statistical concepts and algorithms for an optimal NN search and for finding next neighbors in convex hulls are explained. At the end of this section we analyze the different factors which have strong influence on the retrieval costs. In Section 4 we discuss whether different distance functions can improve the retrieval results. Section 5 concludes this paper.

# Chapter 2

# Statistical Analysis of Distances

In this section we develop the interval distribution, density, expecting values, and the standard deviation of $L_2$-distances in $d$-dimensional space. We assume $d$-dimensional vectors uniformly distributed in $[0..1]^d$. In other words, the values for a given dimension i are uniformly distributed in [0..1].

The $L_k$-norm of a vector v is defined as

$$||\mathrm{v}||_k = (\mathrm{v}^T \mathrm{v})^{\frac{1}{k}} = \sqrt[k]{\sum_{i=1}^{d} |\mathrm{v}_i|^k}$$

The $L_k$-distance between two vectors v and w is $||\mathrm{v} - \mathrm{w}||_k$. At first we show how the distance between uniformly distributed random values is distributed.

**Theorem 2.1** *The interval distribution of the distance between two uniformly distributed random values ranging in* [0..1] *is*

$$P_{|\text{v-v}|}(X < \mathrm{x}) = \begin{cases} \mathrm{x} < 0 & : \quad 0 \\ 0 <= \mathrm{x} <= 1 & : \quad 2\mathrm{x} - \mathrm{x}^2 \\ \mathrm{x} > 1 & : \quad 1 \end{cases}$$

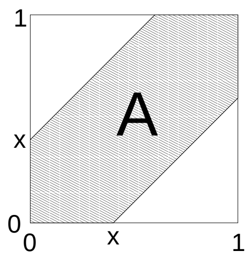


Figure 2.1: Probability as volume over shaded area A

**Proof.** The density function of a uniformly distributed value in [0..1] is $\phi_{\mathrm{V}}(\mathrm{x}) = 1$ and the joint probability density function of two independent and uniformly distributed values in [0..1] is again $\phi_{\mathrm{V},\mathrm{V}}(\mathrm{x}_1, \mathrm{x}_2) = 1$. In order to compute the interval distribution of the absolute difference between the two values we need to compute the volume over the shaded area in Figure 2.1. Since the joint density is 1 for the shaded area it is sufficient to compute the shaded area by subtracting both triangles from 1:

$$\begin{aligned} \mathrm{P}_{|\mathrm{V}\text{-}\mathrm{V}|}(\mathrm{X} < \mathrm{x}) &= \iint\limits_{\mathrm{A}} \phi_{\mathrm{V},\mathrm{V}}(\mathrm{x}_1, \mathrm{x}_2) d\mathrm{x}_1 d\mathrm{x}_2 \\ &= 1 - (1 - \mathrm{x})^2 = 2\mathrm{x} - \mathrm{x}^2 \end{aligned}$$

Therefore, for $\mathrm{x} \in [0..1] : \mathrm{P}_{|\mathrm{V}\text{-}\mathrm{V}|}(\mathrm{X} < \mathrm{x}) = 2\mathrm{x} - \mathrm{x}^2$. Since the maximal distance between two random values is 1 and the minimal distance is 0 we set

$$\mathrm{P}_{|\mathrm{V}\text{-}\mathrm{V}|}(\mathrm{X} < \mathrm{x}) = 1 \text{ for } \mathrm{x} > 1$$

$$\mathrm{P}_{|\mathrm{V}\text{-}\mathrm{V}|}(\mathrm{X} < \mathrm{x}) = 0 \text{ for } \mathrm{x} < 0.$$

□

The density function of the distance between two independent and uniformly distributed random values is defined by the derivation of the interval distribution:

$$\begin{aligned} \phi_{|\mathrm{V}\text{-}\mathrm{V}|}(\mathrm{x}) &= \frac{dP_{|\mathrm{V}\text{-}\mathrm{V}|}(\mathrm{X} < \mathrm{x})}{d\mathrm{x}} = \frac{d(2\mathrm{x} - \mathrm{x}^2)}{d\mathrm{x}} \\ &= 2 - 2\mathrm{x} \text{ for } 0 \leq \mathrm{x} \leq 1 \text{ and } 0 \text{ otherwise} \end{aligned}$$

The expecting value is

$$E_{|\mathrm{V}\text{-}\mathrm{V}|} = \int_{-\infty}^{\infty} \mathrm{x} * \phi_{|\mathrm{V}\text{-}\mathrm{V}|}(\mathrm{x}) d\mathrm{x} = \int_{0}^{1} \mathrm{x} * (2 - 2\mathrm{x}) d\mathrm{x} = \frac{1}{3}$$

and the standard deviation $\sigma_{|\mathrm{V}\text{-}\mathrm{V}|}$ is therefore

$$\sigma_{|\mathrm{V}\text{-}\mathrm{V}|} = \sqrt{\int_{-\infty}^{+\infty} x^2 \phi_{|\mathrm{V}\text{-}\mathrm{V}|}(\mathrm{x}) d\mathrm{x} - E^2_{|\mathrm{V}\text{-}\mathrm{V}|}} = \sqrt{\int_{0}^{1} \mathrm{x}^2 (2 - 2\mathrm{x}) d\mathrm{x} - \frac{1}{9}} = \frac{1}{\sqrt{18}}.$$

For the computation of the distance ($\mathrm{L}_2$) between vectors, the value differences in the respective dimensions are squared. By replacing x with $\sqrt{\mathrm{x}}$ we obtain:

$$\mathrm{P}_{|\mathrm{V}\text{-}\mathrm{V}|^2}(\mathrm{X} < \mathrm{x}) = \begin{cases} \mathrm{x} < 0 & : \ 0 \\ 0 <= \mathrm{x} <= 1 & : \ 2\sqrt{\mathrm{x}} - \mathrm{x} \\ \mathrm{x} > 1 & : \ 1 \end{cases}$$

The density function of the squared distance between two uniformly distributed random values is defined by the derivation of the corresponding interval distribution:

$$\begin{aligned} \phi_{|\mathrm{V}\text{-}\mathrm{V}|^2}(\mathrm{x}) &= \frac{dP_{|\mathrm{V}\text{-}\mathrm{V}|^2}(\mathrm{X} < \mathrm{x})}{d\mathrm{x}} = \frac{d(2\sqrt{\mathrm{x}} - \mathrm{x})}{d\mathrm{x}} \\ &= \frac{1}{\sqrt{\mathrm{x}}} - 1 \text{ for } 0 \leq \mathrm{x} \leq 1;\ 0 \text{ otherwise} \end{aligned}$$

The expecting value is

$$E_{|\text{V-V}|^2} = \int_{-\infty}^{\infty} \text{x} * \phi_{|\text{V-V}|^2}(\text{x})d\text{x} = \int_{0}^{1} \text{x}\left(\frac{1}{\sqrt{\text{x}}} - 1\right)d\text{x} = \frac{1}{6}$$

and the standard deviation $\sigma_{|\text{V-V}|^2}$ is therefore

$$\sigma_{|\text{V-V}|^2} = \sqrt{\int_{-\infty}^{+\infty} x^2\phi_{|\text{V-V}|^2}(\text{x})d\text{x} - E^2_{\text{V-V}^2}} = \sqrt{\int_{0}^{1} \text{x}^2(1/\sqrt{\text{x}} - 1)d\text{x} - \frac{1}{36}} = \sqrt{\frac{7}{180}}.$$

For the distance computation the squared value differences for the corresponding dimensions are added. The computation of the interval distribution is based on the joint density of the sum terms. If we consider only two dimension then we obtain the joint density:

$$\phi_{|\text{V-V}|^2,|\text{V-V}|^2}(\text{x}_1, \text{x}_2) = (\frac{1}{\sqrt{\text{x}_1}} - 1) * (\frac{1}{\sqrt{\text{x}_2}} - 1) \text{ for } \text{x}_1, \text{x}_2 \in [0..1] \text{ and 0 otherwise.}$$

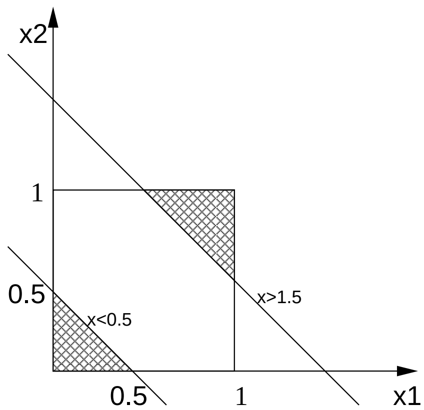


Figure 2.2: Probability as volume over the join density

The probability function of the sum of the two random values is computed over the area $\text{x}_1+\text{x}_2<\text{x}$. Figure 2.2 shows graphically the rectangle spanned by $\text{x}_1$ and $\text{x}_2$ for which the join density is different to zero. Computing $P(X<x)$ with $0\leq x\leq 1$ requires to compute the integral over the shaded triangle in the bottom left corner:

$$\int_{0}^{\text{x}} \int_{0}^{\text{x-x}_2} (\frac{1}{\sqrt{\text{x}_1}} - 1) * (\frac{1}{\sqrt{\text{x}_2}} - 1)d\text{x}_1 d\text{x}_2$$

For values x with 1≤x≤2 we have to compute the difference between 1 and the volume over the right top triangle:

$$1 - \int_{\mathrm{x}-1}^{1} \int_{\mathrm{x}\text{-}\mathrm{x}_2}^{1} (\frac{1}{\sqrt{\mathrm{x}_1}} - 1) * (\frac{1}{\sqrt{\mathrm{x}_2}} - 1) d\mathrm{x}_1 d\mathrm{x}_2.$$

Unfortunately, the analytical computation of the interval distribution is very complex. Even for the 2-dimensional case we obtain very complex formulas. Therefore, we use some lemmas to compute the expecting values and standard deviations.

**Theorem 2.2** *The expecting value of squared $L_2$-distances between uniformly distributed vectors in d-dimensional space $[0..1]^d$ is*

$$E_{||\mathrm{v}\text{-}\mathrm{v}||_2^2} = \frac{d}{6}.$$

**Proof.** We use the following lemma:

> If $X_1, ..., X_i$ are random variables and $E_{X_1}, ..., E_{X_i}$ their corresponding expecting values, then $E_{X_1+...+X_i} = E_{X_1} + ... + E_{X_i}$ holds.

Due to $E_{|\mathrm{v}\text{-}\mathrm{v}|^2} = \frac{1}{6}$ the expecting value of the sum of $d$ squared differences is

$$E_{||\mathrm{v}\text{-}\mathrm{v}||_2^2} = \frac{d}{6}.$$

□

**Theorem 2.3** *The standard deviation of squared $L_2$-distances between uniformly distributed vectors in d-dimensional space $[0..1]^d$ is*

$$\sigma_{||\mathrm{v}\text{-}\mathrm{v}||_2^2} = \sqrt{\frac{7d}{180}}.$$

**Proof.** We use the following lemma:

$$\sigma^2(\sum_{i=1}^{d} a_i X_i) = \sum_{i=1}^{d} a_i^2 * \sigma^2(X_i) + 2 \sum \sum_{i<j} a_i a_j * \mathrm{cov}(X_i X_j)$$

where *cov* represents the covariance. In our case we have to add the squared differences of the respective dimensions. We know, that the dimensions are mutually independent from each other

$$\mathrm{cov}(X_i X_j) = 0 \text{ for } i \neq j$$

and the variances are the same. Applying the rule above the standard deviation $\sigma_{||\text{v-v}||_2^2}$ is therefore

$$\sigma_{||\text{v-v}||_2^2} = \sqrt{\frac{7d}{180}}.$$

□

Following the argumentation in [AHK01] and since the expecting value for $E_{|\text{v-v}|^2}$ is $\frac{1}{6}$ we can write

$$\frac{||\text{x}_1 - \text{x}_2||_2^2}{d} \rightarrow_p \frac{1}{6}$$

where $\text{Y}_d \rightarrow_p c$ means that a vector sequence $\text{Y}_1, ..., \text{Y}_d$ converges in probability to a constant vector $c$ if: $\forall \epsilon > 0 \lim_{d\rightarrow\infty} \text{P}[|\text{Y}_d - c| \leq \epsilon] = 1$. Using the Slutsky's theorem[1] we obtain

$$||\text{x}_1 - \text{x}_2||_2 \rightarrow_p \sqrt{\frac{d}{6}}.$$

Experiments with a high number of dimensions confirm this result. For a low number of dimensions the expecting value differs from $\sqrt{\frac{d}{6}}$. Especially if $d = 1$ then the correct expecting value is $\frac{1}{3}$ instead. Figure 2.3 compares experimental results with varying numbers of dimensions with $\sqrt{\frac{d}{6}}$.

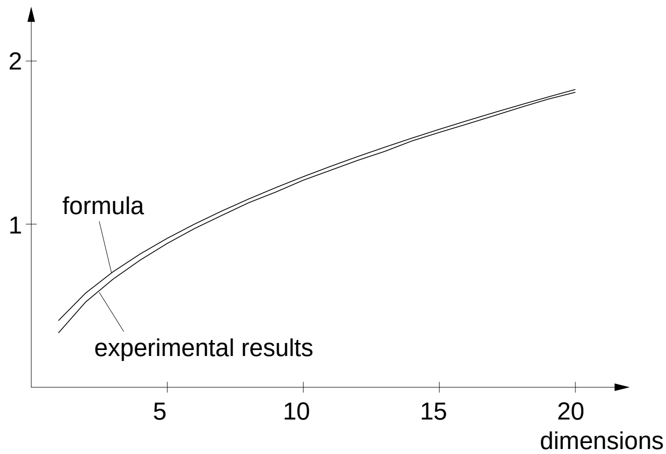


Figure 2.3: Expecting values from experiments with 5000 points converging to $\sqrt{\frac{d}{6}}$

The analytical computation of $\sigma_{||v-v||_2}$ is very hard. Experiments with a high number of dimensions, however, show that the standard deviation of $\text{L}_2$-distances converges fast towards the constant 0.24 (cf. Figure 2.4). This result is confirmed by the results of [AHK01] which shows that the expecting value of the difference between the maximal and minimal $\text{L}_2$-distance to the null-vector converges to a constant independently from the number of dimensions. Table 2.1 summarizes the results found so far.

[1] Slutsky's Theorem: Let $\text{Y}_1...\text{Y}_d...$ be a sequence of random vectors and $h(.)$ be a continuous function. If $\text{Y}_d \rightarrow_p c$ then $h(\text{Y}_d) \rightarrow_p h(c)$.

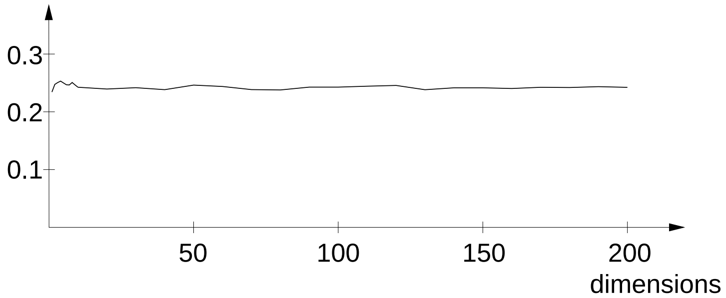


Figure 2.4: Standard deviation of $L_2$-distances for different numbers of dimensions

| X | P(X<x) | $\phi$(x) | E(X) | $\sigma$ |
|---|---|---|---|---|
| $\lvert$v-v$\rvert$ | 2x-x$^2$ | 2-2x | $\frac{1}{3}$ | $\frac{1}{3\sqrt{2}}$ |
| $\lvert$v-v$\rvert^2$ | 2$\sqrt{x}$-x | $\frac{1}{\sqrt{x}} - 1$ | $\frac{1}{6}$ | $\sqrt{\frac{7}{180}}$ |
| $\lvert\lvert$v-v$\rvert\rvert_2^2$ | – | – | $\frac{d}{6}$ | $\sqrt{\frac{7d}{180}}$ |
| $\lvert\lvert$v-v$\rvert\rvert_2$ | – | – | $\rightarrow_p \sqrt{\frac{d}{6}}$ | $\rightarrow_p c$ |

Table 2.1: Summarized results

In the following sections we will show the importance of the ratio $pdist_{far}/pdist_{nn}$ and the difference $pdist_{far} - pdist_{nn}$. $pdist_{far}$ ($pdist_{nn}$) is the distance between a query point and its nearest (furthest) neighbor from a given set of points. We already know that the distance difference $pdist_{far} - pdist_{nn}$ converges very fast to a constant value.

Following the Chebyshev's theorem, the ratio between the nearest and the farthest neighbors can be expressed by introducing $\mu_d$ (the expecting value E(X)), $\sigma_d$, $k_{far}$, and $k_{nn}$:

$$\frac{pdist_{far}}{pdist_{nn}} = \frac{\mu_d + k_{far} * \sigma_d}{\mu_d - k_{nn} * \sigma_d} = \frac{1 + \frac{k_{far}*\sigma_d}{\mu_d}}{1 - \frac{k_{nn}*\sigma_d}{\mu_d}}$$

We know that if the number of dimensions goes towards infinity the mean value $\mu_d$ for $L_2$ distances tends towards infinity, too, whereas the standard deviation $\sigma_d$ remains stable. Therefore, the ratio between maximal distance and minimal distance is decreasing and tends towards 1[2]:

$$\lim_{d\to\infty} \frac{pdist_{far}}{pdist_{nn}} = \lim_{d\to\infty} \frac{1 + \frac{k_{far}*\sigma_d}{\mu_d}}{1 - \frac{k_{nn}*\sigma_d}{\mu_d}} = 1.$$

The key term of the formula above is the ratio $\frac{\sigma_d}{\mu_d}$ which converges with increasing number of dimensions towards zero.

[2]This result is confirmed in [BGRS99].

| | |
|---|---|
| $d$ | number of dimensions |
| $p = \{p_1, p_2, \ldots, p_n\}$ | set of $n$ $d$-dimensional points with $p_i \in [0..1]^d$ |
| $q \in [0..1]^d$ | query point |
| $pdist_d(p_i, p_j)$ | distance function between two $d$-dimensional points |
| $p_{nn} \in p$ | nearest neighbor of $q$: $\forall p_j \in p : pdist_d(q, p_{nn}) \leq pdist_d(q, p_j)$ |
| $p_{far} \in p$ | farthest neighbor of $q$: $\forall p_j \in p : pdist_d(q, p_{far}) \geq pdist_d(q, p_j)$ |
| $pdist_{nn}$ | nearest neighbor distance $pdist_{nn} = pdist_d(q, p_{nn})$ |
| $pdist_{far}$ | farthest neighbor distance $pdist_{far} = pdist_d(q, p_{far})$ |
| $c = \{c_1, c_2, \ldots, c_{cn}\}$ | point set $p$ partitioned into a set of $cn$ $d$-dimensional clusters |
| $points(c_i) \subseteq p$ | points of cluster $c_i \in c$: with $i \neq j \rightarrow points(c_i) \cap points(c_j) = \emptyset$ and $\bigcup_k points(c_k) = p$ |
| $cdist_{nn}(q, c_i)$ | distance between query point $q$ and nearest neighbor of a cluster: $cdist_{nn}(q, c_i) = pdist_d(q, p_{c_i,nn})$ and $p_{c_i,nn} \in points(c_i)$ where $\forall p_j \in points(c_i) : pdist_d(q, p_{c_i,nn}) \leq pdist_d(q, p_j)$ |
| $cdist_d(q, c_i)$ | distance between query point $q$ and nearest (implicit) point on the border of a cluster $c_i$ with $cdist_d(q, c_i) \leq cdist_{nn}(q, c_i)$ |
| $error_d(q, c_i)$ | the approximation error for a given query $q$ and a cluster $error_d(q, c_i) = cdist_{nn}(q, c_i) - cdist_d(q, c_i)$ |

Table 3.1: Notations and basic definitions

# Chapter 3

# Relative Retrieval Costs

In this section we will find an estimation for the costs of the retrieval of the nearest neighbor in dependence on the number of dimensions. Before we start to discuss this problem we introduce some notations and definitions in Table 3.1. For the sake of simplicity we assume only one level of clusters. The common approach is to (i) sort the clusters by their cluster distances $cdist_d$ to the query point. (ii) The nearest neighbor distances $cdist_{nn}$ of the clusters are then determined in the order of their cluster distances as long as the cluster distance of the current cluster is smaller than the nearest neighbor found so far (closest point candidate). After termination the closest point candidate is

**General nearest neighbor algorithm**

**input:** $q, c = \{c_1, ..., c_{cn}\}, d$
**output:** $p_{nn}$
**variables:**
double distance, mindist = $d$, cdist[1,..,$cn$];
double cindex[1,..,$cn$]; // cluster IDs
point candidate;
int index = 1;
**step 1:** // initialization
for (int i = 1; i $\leq$ $cn$; i++){
  cdist[i] = $cdist_d(q, c_i)$;
  cindex[i] = i;}
sort(cindex,cdist); // sort index array with
  // increasing distances
**step 2:**
while (cdist[cindex[index]] < mindist && index$\leq$ $cn$){
  distance = $cdist_{nn}(q, c_{\text{cindex[index]}})$;
  if (distance < mindist){
    mindist = distance;
    candidate = $p_{c_i,nn}$;}
  index++;}
return candidate;

Figure 3.1: General nearest neighbor algorithm

the nearest neighbor. Figure 3.1 shows this algorithm.

An optimal algorithm scans only clusters of which the cluster distances are smaller than $pdist_{nn}$. Since we want to find a lower bound for the cost we assume the $pdist_{nn}$ value to be known in advance which, of course, does not reflect the reality. Figure 3.2 shows the optimal algorithm resulting from a small modification of the previous algorithm. This algorithm is similar to the algorithm *NN-opt* presented in [BBKK97].

We assume that query points have the same distribution as the data points. Furthermore, we assume that the cost for computing the cluster distances and their sorting (step 1) is relatively small and fixed with respect to the number of dimensions whereas the cost for the second step is essential for the retrieval cost. The relative cost in comparison with scanning all clusters and computing their $cdist_{nn}$ values (linear complexity) is therefore at least

$$relcost \geq \frac{|\{c_i \in c | cdist_d(q, c_i) < pdist_{nn}\}|}{cn}.$$

**Optimal nearest neighbor algorithm**

**input:** $q, c = \{c_1, ..., c_{cn}\}, pdist_{nn}, d$
**output:** $p_{nn}$
**variables:** ...
**step 1:** ...
**step 2:**
while (cdist[cindex[index]] $< pdist_{nn}$ && index$\leq cn$){
...

Figure 3.2: Optimal nearest neighbor algorithm (changed lines only)

The low efficiency of high-dimensional indexing is caused primarily by the difference between $cdist_{nn}(q, c_i)$ and $cdist_d(q, c_i)$. This distance is the result of the cluster approximation. Cluster approximation is needed to compute very fast the cluster distances for step 1 in order to reduce the number of clusters to be checked in step 2.

Using the definition of the approximation error we obtain

$$\begin{aligned} relcost &\geq \frac{|\{c_i \in c | cdist_{nn}(q, c_i) - error_d(q, c_i) < pdist_{nn}\}|}{cn} \\ &\geq \frac{|\{c_i \in c | cdist_{nn}(q, c_i) - pdist_{nn} < error_d(q, c_i)\}|}{cn}. \end{aligned}$$

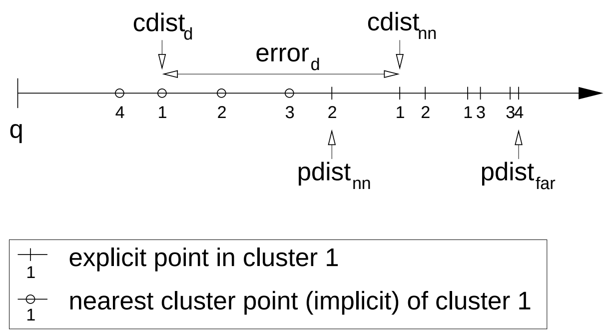


Figure 3.3: Distances from a query point q

Figure 3.3 demonstrates the distances between points and clusters from a given query point q. The high approximation error between the distance to the point in cluster 1 $({}^{|}_{1})$ and the nearest distance to its cluster $({}^{\circ}_{1})$ causes a cluster intersection with the NN-sphere. Following this graphical model we generalize the problem by using statistical concepts and obtain:

$$relcost \geq P(cdist_{nn}(q, c_i) - pdist_{nn} < error_d(q, c_i)).$$

## 3.1 Convex Hull

Since a convex hull cluster is the smallest possible convex cluster shape we base our cost estimation on convex hull clusters. An efficient computation of the $cdist_d$ values for a convex hull cluster in high-dimensional space is not simple and convex hulls should, therefore, not be used in practice. However, since a convex hull is a *minimal convex cluster shape* we use it as theoretical and experimental basis in order to estimate the lower bound of retrieval costs.

We will show that the problem of high-dimensional indexing is caused by the growth of the approximation error $error_d(q, c_i)$ whereas the difference $cdist_{nn}(q, c_i) - pdist_{nn}$ does not increase accordingly.

### 3.1.1 Growth of the Approximation Error

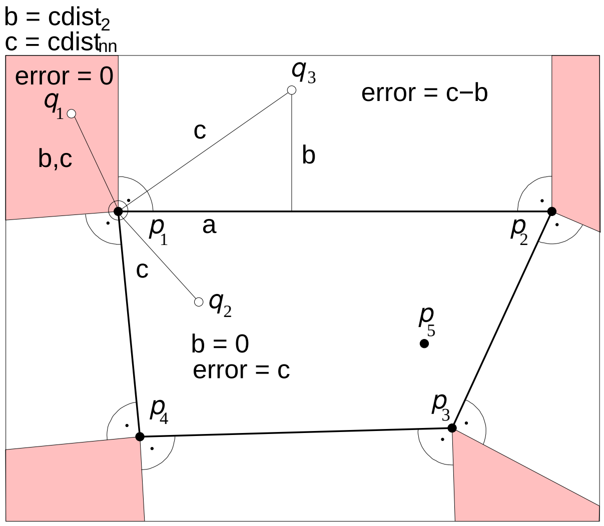


Figure 3.4: Error estimation for a convex hull cluster

Figure 3.4 shows a convex hull cluster of 5 points in a plane. In general, we can distinguish between 3 different areas:

1. *inside area:* The inside area refers to the cluster points within the convex hull cluster. A convex hull cluster $c_i$ is defined on a set of data points $points(c_i)$ called *explicit cluster points* whereas it consists of all points $p$ (called *cluster points*) for which a convex combination exists:

$$inside(\{p_1, p_2, ..., p_n\}) =$$
$$\{p \in [0..1]^d | \exists \lambda_1, \lambda_2, ..., \lambda_n \in [0..1] : p = \sum_{i=1}^{n} \lambda_i * p_i \wedge \sum_{i=1}^{n} \lambda_i = 1\}.$$

The distance of a query point within the inside area, e.g. $q_2$, to the nearest cluster point is always zero. Therefore, the error is equal to the distance to the nearest explicit cluster point.

2. *corner area:* The corner area is defined by all points $q$ for which a point $p_{pivot}$ inside the convex hull exists that for all convex hull points $p$ except $p_{pivot}$ the angle between $\overline{p_{pivot}q}$ and $\overline{p_{pivot}p}$ is greater than $\pi/2$. For the angle computation we use the scalar product $\circ$:

$$corner(\{p_1, p_2, ..., p_n\}) =$$
$$\{q \in [0..1]^d | \exists p_{pivot} \in inside(\{p_1, p_2, ..., p_n\}) : (\forall p \in inside(\{p_1, p_2, ..., p_n\}) :$$
$$p = p_{pivot} \vee \frac{(q - p_{pivot}) \circ (p - p_{pivot})}{pdist_d(q, p_{pivot}) * pdist_d(p, p_{pivot})} < 0)\}.$$

The corner area of the example in Figure 3.4 consists of all the shaded areas.

From this definition it is obvious that for every query point, e.g. $q_1$, within the corner area the nearest cluster point coincides always with the nearest explicit cluster point resulting in an exact approximation ($error = 0$). Furthermore, the corner area and the inside area are disjoint:

$$inside(\{p_1, p_2, ..., p_n\}) \cap corner(\{p_1, p_2, ..., p_n\}) = \emptyset.$$

3. *outside area:* The outside area consists of all points which belong neither to the inside nor to the corner area:

$$outside(\{p_1, p_2, ..., p_n\}) = [0..1]^d \setminus (corner(\{p_1, p_2, ..., p_n\}) \cup inside(\{p_1, p_2, ..., p_n\})).$$

The error for a query point in that area, e.g. $q_3$, is equal to the difference between the nearest explicit cluster point and the nearest cluster point.

We will show that the error grows as the number of dimensions increases. The error is equal to zero only if the query point lies within the corner area.

**Theorem 3.1** *As the number of dimensions increases the probability of a query point being in the corner area of a convex hull cluster tends towards zero if the query point and the data points are uniformly distributed in* $[0..1]^d$.

**Proof.** A query point is distributed in the same way as the data points are distributed. Therefore, the distances between a query point $q$ in the corner area and all explicit cluster points follow the same probability as the distance between two arbitrary uniformly distributed data points. The distance between a query point $q$ within the corner area and an explicit cluster point $p$ which is not $p_{pivot}$ must follow the same distribution. Here we show that this cannot be guarantied in high dimensional space. This distance is minimal if the angle between $\overline{p_{pivot}p}$ and $\overline{p_{pivot}q}$ is minimal that is the angle is near to the right angle. Therefore, we can write for every query point $q$ and every explicit cluster point $p$ which is not the nearest explicit cluster point $p_{pivot}$:

$$pdist_d(q,p) > \sqrt{pdist_d(q,p_{pivot})^2 + pdist(p_{pivot},p)^2}.$$

In an extreme case, the distances $pdist_d(q,p_{pivot})$ and $pdist_d(p_{pivot},p)$ are equal to $pdist_{nn}$ and $pdist_d(q,p)$ is equal to $pdist_{far}$. Since $pdist_{nn}$ is the smallest and $pdist_{far}$ is the farthest distance we can conclude from the previous inequality the following:

$$pdist_{far} > \sqrt{2 * pdist_{nn}^2} = \sqrt{2} * pdist_{nn}$$

and

$$\frac{pdist_{far}}{pdist_{nn}} > \sqrt{2}.$$

We know that for increasing number of dimensions the ratio $pdist_{far}/pdist_{nn}$ converges towards 1 and becomes smaller than $\sqrt{2}$. Therefore, the probability of a query point being in the corner area tends towards zero as the number of dimensions increases. □

Experiments show that the probability of a query point belonging to the corner area converges very fast to zero. Before this can be shown it is necessary to describe the algorithm used for the experiments to compute $cdist_d$ between a query point $q$ and a convex hull cluster. We use here a greedy algorithm to compute a value near to $cdist_d$.

### Cluster Distance Algorithm for Convex Hulls

The main idea of this algorithm is to use a point cursor which always lies within a convex hull cluster and to move it incrementally towards the query point $q$. In a first step the cursor is set to the nearest explicit cluster point $p_1$. Then for each further explicit cluster point $p_2$ the algorithm checks if a point on the line between the cursor $p_1$ and $p_2$ is nearer to the query point $q$ than the cursor itself. If there is a nearer point $p$ then the cursor is set to this point. The algorithm computes the scalar product to update the cursor. After all explicit cluster points were checked the algorithm repeats the procedure for every explicit cluster point again. The input variable *accuracy* defines the number of loops. The greater the value of *accuracy* the nearer is the cursor to the nearest cluster point. The value of *accuracy* is determined by a trade-off between accuracy and computation time.

Figure 3.6 depicts the idea behind the algorithm. The explicit cluster point $p_2$ needs to be checked whether it helps to find a cluster point nearer to $q$ or not. At first, the value of $\lambda$ is computed by using the scalar product:

$$\lambda = \frac{(q-p_1) \circ (p_2 - p_1)}{b^2}$$

If $(0 < \lambda \leq 1)$ then we set the cursor to a nearer cluster point by using a linear combination of $p_1$ and $p_2$:

$$cursor_{new} = \lambda * p_2 + (1-\lambda) * p_1$$

We ran experiments with 5000 data points, 10 convex clusters, 100 query points, and *accuracy*=10 and obtained the relative frequencies of query points being in the corner, in the inside, and in the outside area as shown in Fig. 3.5.

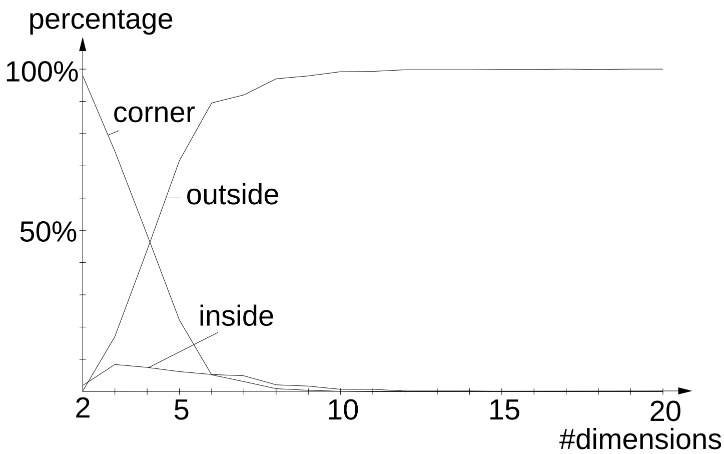


Figure 3.5: Areas

In high-dimensional space almost every query point lies within the outside or in the inside area. If a query points lies within the inside area then the error is at least $pdist_{nn}$ and grows accordingly. The outside case is more tricky as the proof for the following theorem shows:

**Theorem 3.2** *If the query points and the data points in* $[0..1]^d$ *are uniformly distributed then the error of a convex hull approximation for an outside query point is:*

$$error_d(q, c_i) \geq pdist_{nn} * \left(1 - \sqrt{1 - \left(1 - \frac{min(pdist_{far}, \sqrt{2} * pdist_{nn})^2}{2 * pdist_{nn}^2}\right)^2}\right)$$

**Proof.** If a query point $q$ lies within the outside area then there exists at least one explicit cluster point $p_2$ in addition to the nearest explicit cluster point $p_1$ where the nearest point $p$ on the line between $p_1$ and $p_2$ is nearer to $q$ than $p_1$. This follows from the definition of the outside area. Since $p$ is the nearest point on the line the angle between $\overline{pq}$ and $\overline{pp_1}$ is a right angle (cf. Figure 3.6). The approximation error is at least the difference between $a = pdist_d(q, p_1)$ and $x = pdist_d(q, p)$. It is obvious that the nearest cluster point cannot be further away from $q$ than the point $p$ in our triangle:

$$error_d(q, c_i) \geq a - x$$

We can replace $a$ with $pdist_{nn}$ and $x$ with $pdist_{nn} * \sin\alpha$ without violating the inequality:

$$error_d(q, c_i) \geq pdist_{nn} * (1 - \sin\alpha)$$

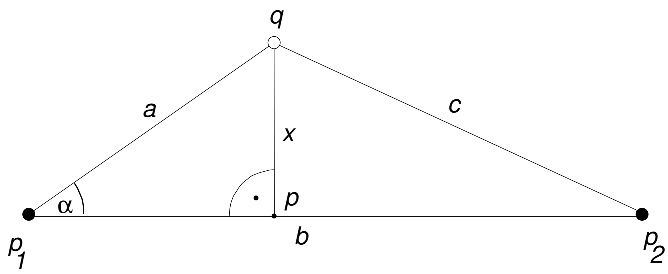


Figure 3.6: Approximation triangle for an outside query point

The angle $\alpha$ can be computed by applying the rule:

$$\cos\alpha = \frac{a^2+b^2-c^2}{2ab}$$

Due to the outside area definition the angle $\alpha$ is in the range between 0 and $\frac{\pi}{2}$:

$$\alpha \in (0..\frac{\pi}{2}).$$

The angle $\alpha$ is maximal if $a$ and $b$ are minimal and $c$ is maximal but not greater than $\sqrt{a^2+b^2}$. The minimal and maximal distances are $pdist_{nn}$ and $\min(pdist_{far}, \sqrt{2} * pdist_{nn})$, respectively. Without violating the inequality we substitute $a, b$, and $c$:

$$\cos\alpha_{max} = \left(1 - \frac{\min(pdist_{far}, \sqrt{2} * pdist_{nn})^2}{2 * pdist_{nn}^2}\right).$$

Applying the rule $\sin^2\alpha_{max} + \cos^2\alpha_{max} = 1$
we obtain

$$\sin\alpha_{max} = \sqrt{1 - \left(1 - \frac{\min(pdist_{far}, \sqrt{2} * pdist_{nn})^2}{2 * pdist_{nn}^2}\right)^2}$$

and the formula:

$$error_d(q, c_i) \geq pdist_{nn} * \left(1 - \sqrt{1 - \left(1 - \frac{\min(pdist_{far}, \sqrt{2} * pdist_{nn})^2}{2 * pdist_{nn}^2}\right)^2}\right)$$

□

If the number of dimensions increases then the ratio $pdist_{far}/pdist_{nn}$ converges to 1 and becomes smaller than $\sqrt{2}$ and we obtain

$$error_d(q, c_i) \geq pdist_{nn} * \left(1 - \sqrt{1 - \left(1 - \frac{pdist_{far}^2}{2 * pdist_{nn}^2}\right)^2}\right).$$

Since the ratio becomes smaller with increasing dimensions the term

$$\left(1 - \frac{pdist_{far}^2}{2 * pdist_{nn}^2}\right)^2$$

grows. Therefore, the factor

$$1 - \sqrt{1 - \left(1 - \frac{pdist_{far}^2}{2 * pdist_{nn}^2}\right)^2}$$

grows, too. Both factors of the cost formula are increasing and therefore the lower bound of the **approximation error grows at least linearly as $pdist_{nn}$ increases.**

Since the ratio $pdist_{far}/pdist_{nn}$ converges to 1 the right factor converges to

$$\lim_{d\to\infty} 1 - \sqrt{1 - \left(1 - \frac{\min(pdist_{far}, \sqrt{2} * pdist_{nn})^2}{2 * pdist_{nn}^2}\right)^2} = 1 - \sqrt{3}/2.$$

This means the length of the triangle sides converge to the same value in high-dimensional space resulting in angles of 60°.

If we run our experiments with 5000 data points and use the $pdist_{nn}$ and $pdist_{far}$ values in order to see the development of the lower bound we obtain the results shown in Figure 3.7.

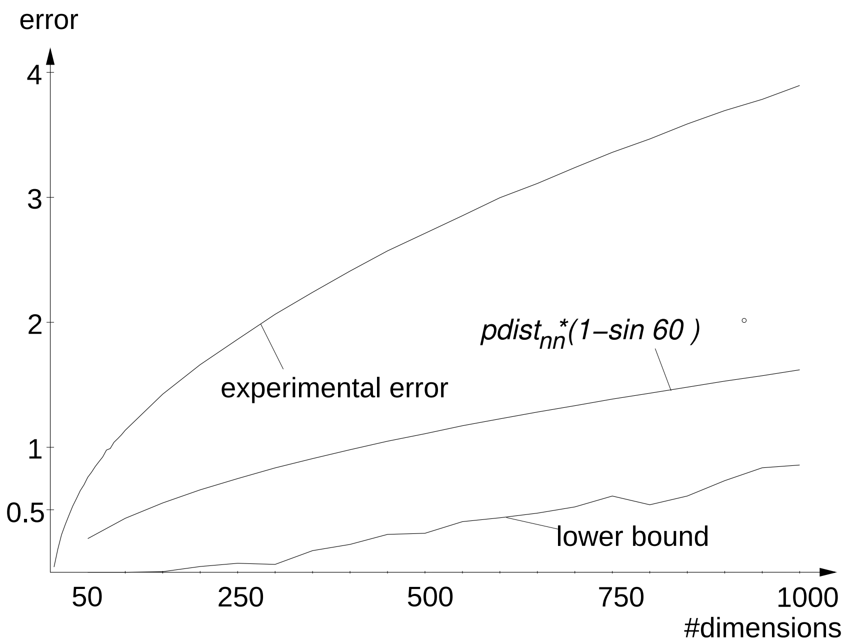


Figure 3.7: $error_d$ and lower bound

We ran further experiments to test the actual error development and compare it with the lower bound. We used the cluster distance algorithm for convex hulls with 5000 data points, 10 convex clusters, 100 query points, and $accuracy$=10 and obtained results displayed in Figure 3.7.

In this subsection we have seen that the error grows at least in proportion to $pdist_{nn}$. We found a growing lower bound for that error. Experiments revealed, however, that the actual error is much higher than the lower bound.

### 3.1.2 Stable Distance Differences

In this subsection we want to show the development of the difference $cdist_{nn}(q, c_i) - pdist_{nn}$ as the number of dimensions increases. In the worst case $cdist_{nn}(q, c_i)$ is $pdist_{far}$:

$$pdist_{far} - pdist_{nn}$$

We know from Section 2 that the standard deviation $\sigma_d$ is stable and that the difference converges to a constant.

Experiments with 5000 data points, 10 convex clusters, 100 query points, and $accuracy$=10 show the following development of the average difference diff=$cdist_{nn} - pdist_{nn}$ and maxdiff=$pdist_{far} - pdist_{nn}$ (cf. Figure 3.8).

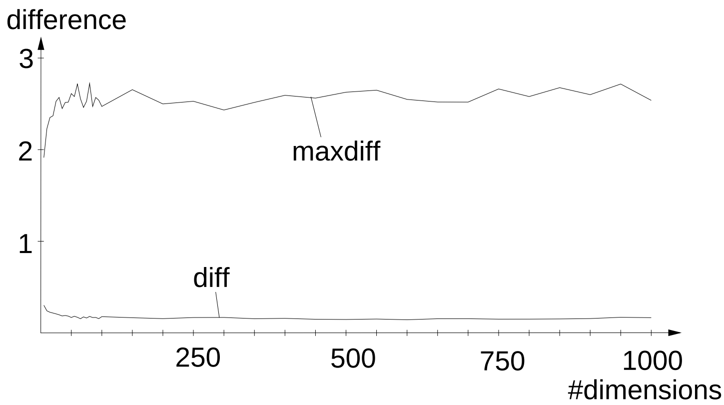


Figure 3.8: Differences

## 3.2 Relative Retrieval Costs

In the begin of this section we developed a lower bound for the relative retrieval cost:

$$relcost \geq P(cdist_{nn}(q, c_i) - pdist_{nn} < error_d(q, c_i)).$$

As the number of dimensions increases the error grows at least linearly as $pdist_{nn}$ whereas the difference $cdist_{nn}(q, c_i) - pdist_{nn}$ remains stable. This means the probability

$$P(cdist_{nn}(q, c_i) - pdist_{nn} < error_d(q, c_i))$$

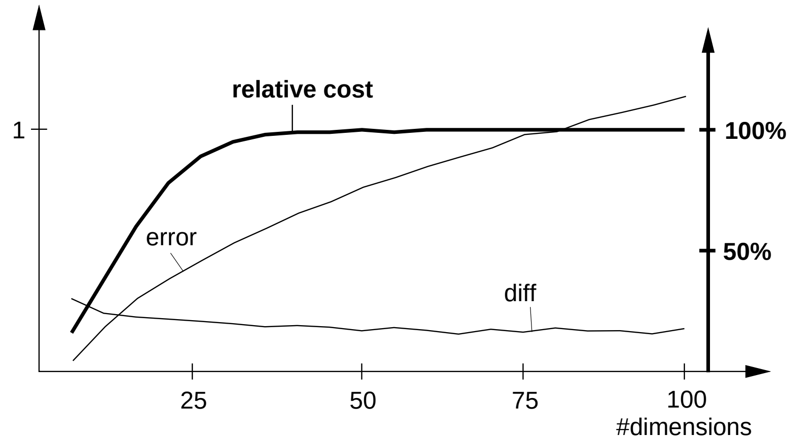


Figure 3.9: Experimental differences and errors

tends towards 100% which causes retrieval costs which are not better than a linear scan without clustering.

Figure 3.9 combines the experimental results. The results show a relative cost near to 100% if the number of dimensions becomes greater than 25. Furthermore, if the error equals the distance difference then we obtain 50% for the relative cost. In other words, the experiments confirm our arguments.

The most interesting question now is: *Which factors influence the relative costs and how can they be used to get a better result?*

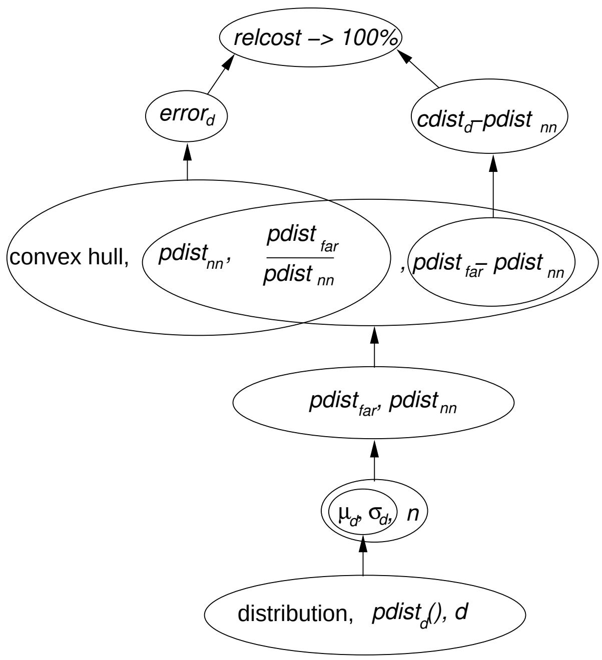


Figure 3.10: Influential factors for the relative cost

Figure 3.10 shows the different factors and their relationships. The values $\mu_d$ and $\sigma_d$ are directly dependent on the data distribution (distribution), the number of dimensions ($d$), and the distance function ($pdist_d()$). These factors together with the number of points ($n$) determine the maximal and the minimal distance ($pdist_{far}, pdist_{nn}$) between points. Our discussion is based on a convex hull cluster. This factor together with the development of $pdist_{far}$ and $pdist_{nn}$ is responsible for the growth of the error ($error_d$) and the stable upper bound for the distance difference ($cdist_d - pdist_{nn}$). Both factors are in turn responsible that the relative costs converges to 100% as the number of dimensions increases.

The number of clusters does not play an important role in the cost estimation since we based our calculation on the probability that for a cluster the approximation error is greater than the distance difference. Of course, if the number of clusters is near to the number of data points then our algorithm is not very efficient. In our discussion we ignored this degenerated case.

Cluster algorithms determine which data points should belong to a cluster. Obviously, cluster algorithms influence the difference $cdist_{nn} - pdist_{nn}$. A good cluster algorithm groups data points with a high space density together (cf. [HK99]. Unfortunately, due to the $\mu_d$ and $\sigma_d$ development the benefit of a good cluster algorithm vanishes in high-dimensional space. The distances between two arbitrary data points become very similar. Furthermore, in our cost estimation we used the maximal difference $pdist_{far} - pdist_{nn}$ which is independent from cluster algorithms. For these reasons we neglected a discussion about the cluster algorithm in high-dimensional space.

The development of the relative cost (cf. Figure 3.10) is heavily based on the factors *cluster shape, number of data points*, and *mean distance and standard deviation.*

## Cluster Shape

We used here the convex hull as the smallest convex cluster shape for a given set of points. The convex hull itself is not an appropriate cluster shape due to the lack of efficient algorithms in high-dimensional space. But by using the convex hull we were able to explain and demonstrate the curse of dimensionality. Therefore, we can conclude that *no convex cluster shape shows better results than the convex hull.* What about non-convex clusters? So far, there are no research results about non-convex clusters known to the author. The general problem of non-convex cluster shapes is to find a formal description of a concrete cluster which needs less description data than for the description of the included explicit data points themselves and which computes $cdist_d$ sufficiently fast and exact.

## Number of Data Points (n)

The data space grows exponentially as the number of dimensions increases. As long as the number of data points does not increase accordingly the space is sparsely populated.

In most practical applications the number of data points is extremely low in comparison with the exponentially growing space and cannot be manipulated to improve retrieval costs.

## Mean Distance and Standard Deviation ($\mu_d, \sigma_d$)

The general problem is the growth of $\mu_d$ whereas $\sigma_d$ does not increase accordingly. These phenomena are caused by the factors distribution of data points, distance function ($pdist_d()$), and number of dimensions ($d$).

### Distribution of Data Points

Here we assumed uniformly distributed data points. In reality, however, the distribution is usually quite different from the uniform distribution. Therefore, depending on the applications the underlying distribution can reduce the retrieval cost, e.g. if the number of 'active' dimensions is low.

Unfortunately, the distribution is application-dependent and can therefore not be manipulated in order to obtain better retrieval results. We know that the values of $\mu_d$ are $\sigma_d$ have a very strong influence on the retrieval cost. If these values for an application are known then we can assess the retrieval costs. Convex hull experiments with these statistical values together with formula from Theorem 3.2 and $pdist_{far} - pdist_{nn}$ can relatively easy compute a lower bound for the relative retrieval costs.

### Distance Function ($pdist_d()$)

In our discussion we used the Euclidean distance function. This function is one of the most often used distance functions for nearest-neighbor search in high-dimensional space. There are many further distance functions but the distance function is usually dependent on the application. In multimedia application the distance function must properly reflect the user's perception of similarity.

At the same time the distance function influences the development of $\mu_d$ and $\sigma_d$. [AHK01] shows that the relative contrast ($\frac{dmax-dmin}{dmin}$) is much better if we use an $L_k$-metric where $k$ is very small. The problem, however, is that $L_k$-metrics with varying $k$ can find different nearest neighbors for a given query point resulting in changed type of similarity.

A very interesting question is therefore: *For a given distance function can we find another distance function with a better $\mu_d$ and $\sigma_d$ development retaining the same nearest neighbors for every arbitrary query point?* There is an easy example for such a replacement. We can drop the square root from the Euclidean distance function and obtain better $\mu_d$ and $\sigma_d$ values and always the same nearest neighbors. We will investigate this idea in the next section in more detail.

**Number of Dimensions ($d$)**

The number of required dimensions is application dependent. In order to get good retrieval results this value should be as small as possible. Due to the well-known curse of dimensionality there are many approaches to reduce the number of dimensions (e.g. [CM00], FastMap [FL95], Generalized NN Search [HAK00], and GENIMI [Fal96]). [HAK00] tries to find a meaningful distance function with a reduced number of dimensions by analyzing distance distribution w.r.t. a given query point.

Regardless of the dimension reduction, many application still need a high number of dimensions and suffer from the curse of dimensionality.

# Chapter 4

# Manipulation of $\mu_d$ and $\sigma_d$

There are many ways to manipulate $\mu_d$ and $\sigma_d$. A straightforward approach is to manipulate the distance function. We can try to substitute the $L_2$-distance with another function which provides better statistical values. A distance function which returns for every arbitrary query point the same sorted (by distance) point sequence as the $L_2$-metric but more efficiently would be optimal.

**Definition 4.1** *Let $p = \langle p_1,\ p_2,\ \ldots, p_n\rangle$ be a sequence of $n$ distinct points and $sort_{dist}$ be a function which maps a set of $n$ points $p$ together with a given query point $q$ to a sorted sequence $\langle p_{i_1}, p_{i_2}, \ldots, p_{i_n}\rangle$ where $j > k$ implies $dist(q, p_{i_j}) \geq dist(q, p_{i_k})$. A distance function $dist_1$ is said to be a* substitute *of $dist_2$ if $\forall p, q : sort_{dist_1}(p, q) = sort_{dist_2}(p, q)$.*

**Theorem 4.1** *Every substitute $dist_1$ of a given distance function $dist_2$ has the same relative cost for the nearest neighbor search as the original distance function $dist_2$: $relcost_{dist_1} = relcost_{dist_2}$.*

**Proof.** In Figure 3.3 we see that the relative cost is determined by the relative number of nearest (implicit) cluster points $cdist_d$ being nearer to $q$ than $pdist_{nn}$. Since a new distance function as a substitute will retain the same order of points it will find the same (implicit) cluster points as nearest cluster points ($cdist_d$) to $q$. For the same reason, the same number of nearest (implicit) cluster points will remain nearer to $q$ than $pdist_{nn}$ resulting in the same relative cost. □

This result means that a distance substitute cannot reduce the relative cost. Nevertheless, it makes still sense to replace the $L_2$ distance function with its substitute $L_2^2$ distance function in order to save the computation of the square root.

Another approach tries to use another distance function which is not a substitute since it changes the order of nearest neighbors. [AHK01] investigates the $L_k$ distance function with varying $k$. They show good statistical results if k is low ($k$ can also be a fractional number).

Choosing another distance function with 'good' statistical results (e.g. relative contrast) does not guarantee low retrieval costs. In the following we will show that the $L_1$-distance function has the same problem as the $L_2$-distance function.

**Theorem 4.2** *The retrieval cost for nearest-neighbor-search among uniformly distributed points in* $[0..1]^d$ *using the* $L_1$*-distance function converges to 100% as the number of dimensions increases (d).*

**Proof.** Following

$$relcost \geq P(cdist_{nn}(q, c_i) - pdist_{nn} < error_d(q, c_i))$$

we have to compare the development of $cdist_{nn}(q, c_i) - pdist_{nn}$ with $error_d(q, c_i)$ as the number of dimensions increases. From Table 2.1 we know the standard deviation for the 1-dimensional $L_1$-distance $\sigma_{|\text{v-v}|} = \sqrt{\frac{1}{18}}$. Therefore we obtain the standard deviation for $d$ dimensions of $\sqrt{\frac{d}{18}}$. This is confirmed by the result from [AHK01] which showed that the difference between the maximal and minimal distance grows in proportion to $\sqrt{d}$.

Now we will investigate the development of the approximation error $error_d(q, c_i)$. Again, we choose a convex hull as minimal convex cluster shape. The nearest (implicit) cluster point of a cluster $c_0$ is found by linear combinations of the explicit cluster points. A greedy algorithm starts with the nearest explicit cluster point p w.r.t. the query point q and $c_0$ and computes the approximate nearest cluster point r by linearly combining p with another explicit cluster point s:

$$cdist_{nn}(\mathrm{q}, c_0) = \sum_{i=1}^{d} |\mathrm{p}_i - \mathrm{q}_i|$$

$$r = \frac{\mathrm{p+s}}{2} \text{ and } cdist_d(\mathrm{q}, c_0) \leq \sum_{i=1}^{d} |\mathrm{r}_i - \mathrm{q}_i|$$

Next, we show why combining two explicit cluster points generates an implicit cluster point which is more likely to be nearer to than farer from the query point q.

Six[1] different cases can be distinguished for the ith component of p, s, and q:

| case | condition | \|p-q\|-\|r-q\| |
|---|---|---|
| case 1: | $\mathrm{p}_i < \mathrm{s}_i < \mathrm{q}_i$ | +x |
| case 2: | $\mathrm{q}_i < \mathrm{s}_i < \mathrm{p}_i$ | +x |
| case 3: | $\mathrm{s}_i < \mathrm{p}_i < \mathrm{q}_i$ | -x |
| case 4: | $\mathrm{q}_i < \mathrm{p}_i < \mathrm{s}_i$ | -x |
| case 5: | $\mathrm{p}_i < \mathrm{q}_i < \mathrm{s}_i$ | +c |
| case 6: | $\mathrm{s}_i < \mathrm{q}_i < \mathrm{p}_i$ | +c |

[1]Without losing generality we ignored the case when the components have the same value.

Every case occurs with the same probability of $\frac{1}{6}$. In cases 1 and 2, the linear combination of p and s generates a new cluster point nearer (+x) to q than p. Cases 3 and 4 neutralize this effect by generating a point farer away from q by the same amount (-x). The cases 5 and 6, however, are the interesting cases. Since q is nearly in the middle of p and s the linear combination provides a cluster point being averagely nearer to q than p resulting in a positive constant +c. This holds for every dimension. Due to the $L_1$-distance definition the constant $\frac{c}{3}$ is multiplied with the number of dimensions ($\frac{dc}{3}$) providing an estimation of the lower bound of $error_d$. This means that the approximation error grows linearly whereas $cdist_{nn}(q, c_i) - pdist_{nn}$ grows in proportion to $\sqrt{d}$. For that reason the relative costs converges to 100% when the number of dimensions increases. □

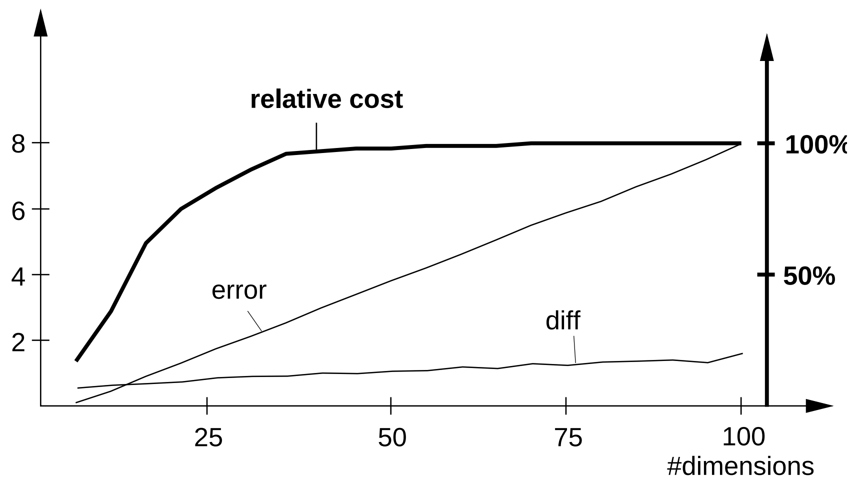


Figure 4.1: Experimental differences and errors

Figure 4.1 compares our experimental results. For the experiments we used the same settings as in the previous section (5000 points, 10 cluster, 100 query points). The nearest cluster point is found by moving a cursor within the convex hull near to the query point. The start point is the nearest explicit cluster point. In every iteration the algorithm finds the best explicit cluster point with a corresponding $\lambda$-value (r=p*(1-$\lambda$)+s*$\lambda$). The input variable *accuracy* defines the number of iterations. The greater the value of *accuracy* the nearer is the cursor to the real nearest cluster point. We used the value 10 for *accuracy*.

An important role for evaluating retrieval costs plays the relative contrast. Instead of using maximal and minimal distances we define the relative contrast on the mean and the standard deviation. In this way, we are independent from the number of data points and it is often easier to compute concrete values:

$$rc_d = \frac{\sigma_d}{\mu_d}$$

Using the relative contrast we can compare different distance functions in correspondence to the results in Section 2:

$$rc_{d,\mathrm{L}_2} = \frac{0.24}{\sqrt{\frac{d}{6}}} = \frac{0.24 * \sqrt{6}}{\sqrt{d}} \approx \frac{0.5878}{\sqrt{d}}$$

$$rc_{d,\mathrm{L}_2^2} = \frac{\sqrt{\frac{7d}{180}}}{\frac{d}{6}} = \frac{\sqrt{\frac{7}{5}}}{\sqrt{d}} \approx \frac{1.1832}{\sqrt{d}}$$

$$rc_{d,\mathrm{L}_1} = \frac{\sqrt{\frac{d}{18}}}{\frac{d}{3}} = \frac{1}{\sqrt{d}\sqrt{2}} \approx \frac{0.7071}{\sqrt{d}}$$

The relative contrast of all three distance functions converge to zero as the number of dimensions increases. But the relative contrast of $\mathrm{L}_1$ and $\mathrm{L}_2$ tends a little faster than $\mathrm{L}_2$ towards zero.

# Chapter 5

# Conclusion

We have focused on explaining the low efficiency of high dimensional nearest neighbor search. In contrast to other works we used a convex hull as minimal convex cluster shape. Therefore, we can state, that any convex cluster shape cannot outperform our results. Furthermore, we related the retrieval costs to the approximation error and to the difference between the farthest and the nearest distance. If we use uniformly distributed data and the $L_2$ distance function then the mean of the distances grows resulting in a high approximation error whereas the standard deviation does not grows accordingly.

The data distribution is given by the application and can therefore not be manipulated to achieve better retrieval results. However, a modification of the distance function seems to be a promising approach. In the last section we discussed whether it makes sense to substitute the $L_2$ distance by the $L_1$ distance function.

In the future we will investigate the impact of further distance functions on the retrieval costs.

# Bibliography


[AHK01] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the Surprising Behavior of Distance Metrics in High Dimensional Space. In *Database Theory — ICDT'01, Proc. of the 8th Int. Conf., London, January 2001*, lncs, Vol. 1973, pages 420–434. Springer-Verlag, Berlin, 2001.

[BBK+01] S. Berchtold, C. Böhm, D. Keim, F. Krebs, and H.-P. Kriegel. On Optimizing Nearest Neighbor Queries in High Dimensional Spaces. In *Database Theory — ICDT'01, Proc. of the 8th Int. Conf., London, January 2001*, lncs, Vol. 1973, pages 435–449. Springer-Verlag, Berlin, 2001.

[BBKK97] S. Berchtold, C. Böhm, D. A. Keim, and H.-P. Kriegel. A Cost Model For Nearest Neighbor Search in High-Dimensional Data Space. In *Proc. of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS'97, Tuscon, Arizona, USA*, pages 78–86. ACM Press, New York, 1997.

[BGRS99] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When Is Nearest Neighbor Meaningful. In C. Beeri and P. Buneman, editors, *Database Theory — ICDT'99, Proc. of the 7th Int. Conf., Jerusalem, Israel, January 1999*, lncs, Vol. 1540, pages 217–235. Springer-Verlag, Berlin, 1999.

[CM00] K. Chakrabarti and S. Mehrotra. Local Dimensionality Reduction: A New Approach to Indexing High Dimensional Spaces. In Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang, editors, *Proc. of the 26th Int. Conf. on Very Large Data Bases, VLDB'00, Cairo, Egypt, September 10–14, 2000*, pages 89–100, Morgan Kaufmann Publishers, 2000.

[Fal96] Christos Faloutsos. *Searching Multimedia Databasees by Content.* Kluwer Academic Publishers, Boston/Dordrecht/London, 1996.

[FBF77] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Transaction on Mathematical Software*, 3(3):209–226, September 1977.

[FL95] Christos Faloutsos and King-Ip Lin. FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets. In Michael J. Carey and Donovan A. Schneider, editors, *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, May 22-25, 1995*, pages 163–174, ACM Press, 1995.

[GG98] V. Gaede and O. Günther. Multidimensional Access Methods. *ACM Computing Surveys*, 30(2):170–231, June 1998.

[HAK00] Alexander Hinneburg, Charu C. Aggarwal, and Daniel A. Keim. What Is the Nearest Neighbor in High Dimensional Spaces? In Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang, editors, *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 506–515, Morgan Kaufmann, 2000.

[HK99] A. Hinneburg and D. A. Keim. Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering. In M. P. Atkinson, M. E. Orlowska, Patrick Valduriez, S. B. Zdonik, and M. L. Brodie, editors, *Proc. of the 25th Int. Conf. on Very Large Data Bases, VLDB'99, Edinburgh, Scotland, UK, September 7–10, 1999*, pages 506–517, Morgan Kaufmann Publishers, 1999.

[Lu99] Guojun Lu. *Multimedia Database Management Systems.* Artech House, Boston/London, 1999.

[WSB98] R. Weber, H.-J. Schek, and S. Blott. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In A. Gupta, O. Shmueli, and J. Widom, editors, *Proc. of the 24st Int. Conf. on Very Large Data Bases, VLDB'98, Ney York City, August 24–27, 1998*, pages 194–205. Morgan Kaufmann Publishers, San Francisco, CA, 1998.