

### Algorithmus nach Korn (2. Fastmap-Algorithmus)

bisherige Baumverfahren und Algorithmen basieren hauptsächlich auf euklidischer Distanz

hier Diskussion der Verfahren für komplexe Distanzfunktionen:

- ♦ keine euklidische Distanzfunktion
- ♦ aufwändig zu berechnen  
(siehe etwa quadratische Distanzfunktionen)

Grundidee: Substitution der komplexen Distanzfunktion (meist euklidische Distanzfunktion)

Anwendung der einfachen Distanzfunktion als Filter

Problem: unterschiedliche Distanzwerte erzeugen Verfälschungen  
→ nach Filtern ist Kontrolle mit richtiger Distanzfunktion erforderlich

Korn, Sidiropoulos, Faloutsos, Siegel, Protopapas 1996

so genanntes GEMINI-Verfahren (Generic Multimedia object Indexing)

folgende Probleme bei Verwendung komplexer Distanzfunktionen werden behoben:

1. aufwändige Berechnung der Distanzen
2. keine effizienten Suchalgorithmen verfügbar
3. Feature-Objekte sind nicht immer Vektoren

gegeben: komplexe Distanzfunktion  $d(o_1, o_2)$   
für beliebige Objektmenge  $O$

Substitution mittels Funktion  $F(o)$  auf Objekten aus  $O$   
(etwa Dimensionsreduzierung) und einfache Distanzfunktion  
 $\delta(F(o_1), F(o_2))$

folgende lower-bound-Bedingung muss erfüllt sein:

$$\forall o_1, o_2 \in O : \delta(F(o_1), F(o_2)) \leq d(o_1, o_2)$$

(garantiert korrekten Ausschluss von Objekten)

Entwurfsziele:

1. effiziente Berechnung wird ermöglicht, etwa RKV-Algorithmus im Baumindex
2. Erfüllung lower-bound-Bedingung
3. minimale Verfälschung

Messung Unähnlichkeit von Zeitreihen anhand euklidischer Distanzfunktion

→ aufwändig durch hohe Anzahl von Messwerten

Dimensionsreduzierung durch DFT-Transformation (Kompaktheit)

Substitution durch euklidische Distanz auf kompakten Fourier-Koeffizienten

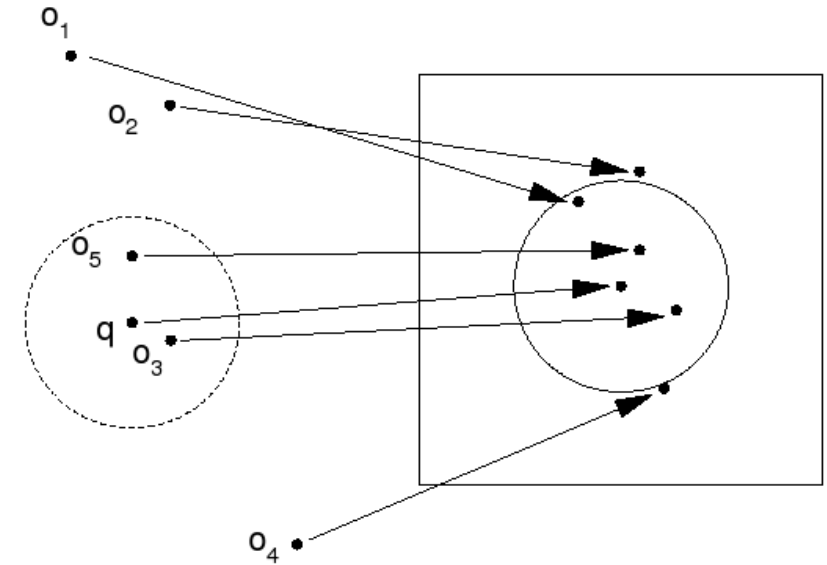
aufgrund Parseval-Theorem und Wegfallen nichtnegativer Summanden ist lower-bound-Bedingung erfüllt

## Algorithmus für Bereichsanfrage nach Korn <is web>

einfache Distanzfunktion  $\delta$  und komplexe Distanzfunktion  $d$   
Transformationsfunktion  $F$   
Anfragepunkt  $q$  und Radius  $r$   
Wurzelknoten  $T$  des Indexbaums

```
[1] procedure Korn-range(punkt q,real radius,funktion d,  
[2]     funktion F,knoten T,objektmenge resultat)  
[3]     range(F(q),radius,T,res) // etwa HS-Algorithmus  
[4]     berechne  $d(q,r)$  für alle  $r$  aus res  
[5]     resultat enthalte alle  $r$  mit  $d(q,r) \leq \text{radius}$   
[6] end procedure
```

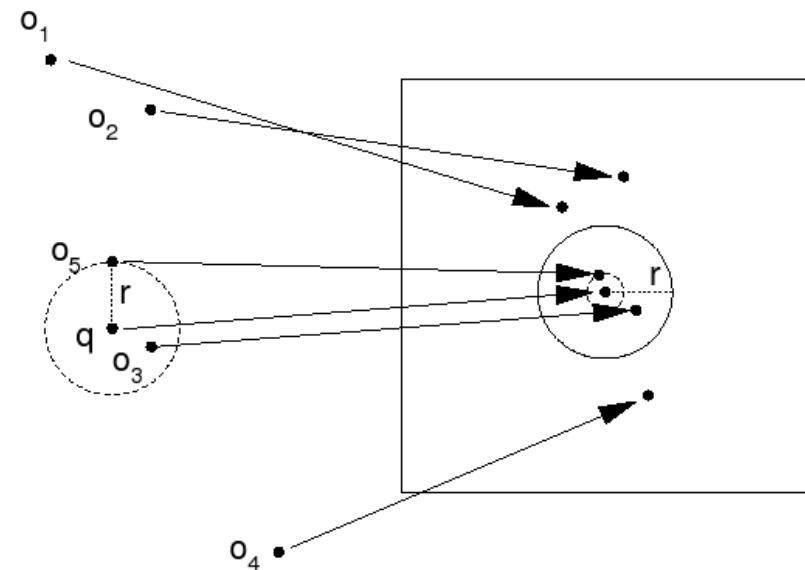
## Algorithmus für Bereichsanfrage graphisch <is web>



## Algorithmus für knn-Anfrage nach Korn <is web>

```
[1] procedure Korn-knn(punkt q,int k,funktion d,  
[2]     funktion F,knoten T,objektmenge resultat)  
[3]     knn(F(q),k,T,resultat) // etwa HS-Algorithmus  
[4]     berechne  $d(q,r)$  für alle  $r$  aus resultat  
[5]     max sei größter  $d(q,r)$ -Wert  
[6]     range(F(q),max,T,neu-resultat) // Bereichssuche  
[7]     berechne  $d(q,nr)$  für alle  $nr$  aus neu-resultat  
[8]     resultat enthalte  $k$  nächste Nachbarn von neu-resultat  
[9] end procedure
```

## Algorithmus für knn-Anfrage graphisch <is web>



Seidl, Kriegel 1998

kritischer Parameter ist  $\max$ -Wert

→ erzeugt u.U. hohe Anzahl von Kandidaten

Idee:  $\max$ -Wert dynamisch ermitteln

→ Verschränkung der Zeilen 3 bis 8 und Einsatz von getNext-Anfrage

```
[1] procedure Seidl-knn(punkt q,int k,funktion d,
[2]     funktion F,knoten T,objektmenge resultat)
[3]     initialisiere getNext(F(q),T) // HS-Algorithmus
[4]     pqueue queue // sortierte Warteschlange
[5]     real max=∞
[6]     while o=getNext(F(q),T) and  $\delta(o,q) \leq \max$  do
[7]         if  $d(o,q) \leq \max$  then enqueue(queue,o,d(o,q))
[8]         if queue.length  $\geq k$  then max=queue[k].distanz
[9]         entferne alle queue-Elemente mit distanz > max
[10]    end do
[11]    resultat enthalte alle queue-Elemente
[12] end procedure
```

Faloutsos, Lin 1995

approximative Abbildung einer Metrik (Objekte und Distanzen)  
auf k-Dimensionale Punkte und  
euklidische Distanzfunktionen

Beispiel: Abbildung Menge von Wörtern und Editierdistanz auf  
mehrdimensionale Punkte

Distanzwerte liegen explizit vor

Parameter k kann frei vordefiniert werden

- ◆ je höher, desto genauer die Approximation
- ◆ je höher, desto mehr Werte pro Objekt

Anwendungen:

Substitution komplexer Distanzfunktionen durch indizierbare Distanzfunktion und Punkte

2D- oder 3D-Visualisierung ( $k=2$  oder  $k=3$ ) einer Metrik

Dimensionsreduzierung

Entwurfsziele sind:

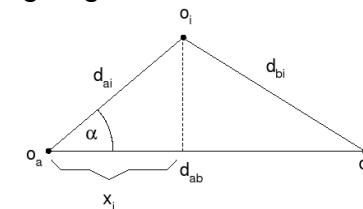
- ◆ effiziente Abbildung
  - ◆ möglichst linearer Aufwand bzgl. Objektanzahl
- ◆ distanzapproximierend
- ◆ effiziente Abbildung neuer Objekte (unabhängig von Objektanzahl)

Grundidee:

- ◆ Objekte liegen bereits (idealisiert) im euklidischen Vektorraum
- ◆ Koordinaten sind jedoch unbekannt

Ziel: Finden orthogonaler Koordinatenachsen und Ermitteln der entsprechenden Werte durch Projektion auf Achsen

Achse ist durch Objektpaar  $o_a$  und  $o_b$  (Pivotpunkte) festgelegt



Anwendung Kosinussatz:

$$x_i = \frac{d_{ai}^2 + d_{ab}^2 - d_{bi}^2}{2d_{ab}} \quad (1)$$

nach Finden der Koordinatenwerte einer Achse müssen Distanzen entsprechend reduziert werden

also Entfernen des entsprechenden Terms aus der euklidischen Distanzfunktion:

$$d'(o_i, o_j)^2 = d(o_i, o_j)^2 - (x_i - x_j)^2 \quad (2)$$

Distanzen werden also kleiner – Problem: negative Werte können auftreten

nach Anpassung

- ♦ Ermitteln der Koordinatenwerte der nächsten Achse, Wiederholung bis k Achsen gefunden wurden

Ziel: Einfluss der Achsendimension soll mit jeder neu gefundenen Achse abnehmen  
daher: Finde die am weitesten auseinander liegenden Objekte

Problem: quadratischer Aufwand

Lösung: Einsatz eines heuristischen Algorithmus mit linearem Aufwand

oa, ob sind Ausgabeparameter!

```
[1] procedure pivot(objektmenge O, funktion d,
[2]     objekt oa, objekt ob)
[3]     wähle beliebiges Objekt aus O als ob
[4]     ermittle oa als das von ob entfernteste Objekt anhand d
[5]     ermittle ob als das von oa entfernteste Objekt anhand d
[6] end procedure
```

Bemerkung: Schritte 4 und 5 werden üblicherweise mehrfach durchlaufen

```

[1] real array [N,k] X //N k-dimensionale Punkte
[2] objekt array [2,k] pivot-objects //k Pivot-Objektpaare
[3] int col=0 //Array-Index
[4]
[5] procedure FastMap(int k,objektmenge O,funktion d)
[6]   if k ≤ 0 then return else col++
[7]   pivot(O,d,oa,ob) //Pivot-Objekte ermitteln
[8]   pivot-objects[1,col]=oa
[9]   pivot-objects[2,col]=ob
[10]  if d(oa,ob)=0 then do
[11]    X[i,col]=0 für alle i=1..N
[12]    return
[13]  end do
[14]  for each objekt oi in O do
[15]    xi ist berechneter Koordinatenwert (siehe Formel 1)
[16]    X[i,col]=xi
[17]  end do
[18]  d' sei modifizierte Distanzfunktion (siehe Formel 2)
[19]  FastMap(k-1,O,d')
[20] end procedure

```

Wörter: Medium (W1), Datenbank (W2), Multimedia (W3), System (W4),  
Objekt (W5)

Distanzen:

	W1	W2	W3	W4	W5
W1	0	8	8	5	6
W2	8	0	10	8	8
W3	8	10	0	8	9
W4	5	8	8	0	6
W5	6	8	9	6	0

Ergebnis des FastMap-Algorithmus:

	W1	W2	W3	W4	W5	Pivot-Punkte
d1	5	10	0	5	5,85	3-2
d2	3,7	0	0	3,7	6,84	3-5
d3	5,04	0	0	2,55	0	3-1
d4	4,34	4,34	4,34	0	4,34	4-1

Annahme: Punkte liegen im (unbekannten) Vektorraum  
Annahme nicht immer erfüllt, da nicht jede Distanzfunktion  
entsprechend einbettbar ist  
Problem bei der Distanzanpassung: Wurzel aus negativen  
Werten

Experimente zeigen: Nächste-Nachbarsuche in Indexbäumen versagt ab etwa 20 Dimensionen

Problem: Ausschluss von Teilbäumen von der Suche nicht möglich  
→ Gesamtbaumdurchlauf aufwändiger als sequentieller Durchlauf

Phänomen wird **Fluch der hohen Dimensionen** genannt

tritt im Wesentlichen bei Verwendung der euklidischen Distanzfunktion auf

Fluch ergibt sich aus der Distanzverteilung

Frage: gilt der Fluch für alle Baumsuchverfahren?

### Quadrierte Distanzverteilung zweier gleichverteilter Werte

Wahrscheinlichkeitsverteilung von Distanzen:

$$f_{|v-v|^2}(x) = \begin{cases} \frac{1}{\sqrt{x}} - 1 & \text{für } 0 \leq x \leq 1 \\ 0 & \text{sonst} \end{cases}$$

$$F_{|v-v|^2}(x) = \begin{cases} 0 & \text{für } x < 0 \\ 1 & \text{für } x > 1 \\ 2\sqrt{x} - x & \text{sonst} \end{cases}$$

Erwartungswert  $\mu_{|v-v|^2}$  beträgt 1/6 und Varianz  $\sigma_{|v-v|^2}^2$  beträgt 7/180

### Quadrierte Distanzverteilung zweier gleichverteilter Punkte

Wahrscheinlichkeitsverteilung der quadrierten eukl. Distanzen nach Anwendung des zentralen Grenzwertsatzes: Annäherung an Normalverteilung

$$\begin{aligned} \lim_{d \rightarrow \infty} F_{||v-v||_2^2}(x) &= \lim_{d \rightarrow \infty} F_{\underbrace{|v-v|^2 + \dots + |v-v|^2}_d}(x) \\ &= \Phi\left(\frac{x - d\mu_{|v-v|^2}}{\sqrt{d}\sigma_{|v-v|^2}}\right) \end{aligned}$$

Erwartungswert

$$\mu_{||v-v||_2^2} = d/6$$

Varianz

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

### Verteilung der eukl. Distanz zweier gleichverteilter Punkte

Wurzelberechnung führt zu

$$\lim_{d \rightarrow \infty} F_{||v-v||_2}(x) = \Phi\left(\frac{x^2 - d\mu_{|v-v|^2}}{\sqrt{d}\sigma_{|v-v|^2}}\right)$$

Erwartungswert  $\mu_{||v-v||_2}$  strebt gegen Wert  $\sqrt{d/6}$  mit steigender Dimension

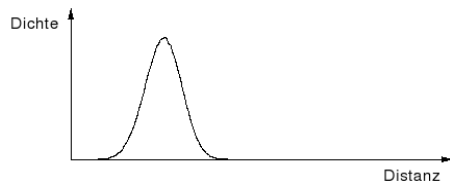
Standardabweichung ist konstant ( $\approx 0,24$ )

### Verteilung der euklidischen Distanz zweier gleichverteilter Punkte (2)

Fazit: im Gegensatz zur Standardabweichung steigt Erwartungswert mit Dimensionszahl (gilt auch für viele Realdatenverteilungen)

### Distanzverteilung im hochdimensionalen Raum graphisch

10 Dimensionen:



100 Dimensionen:



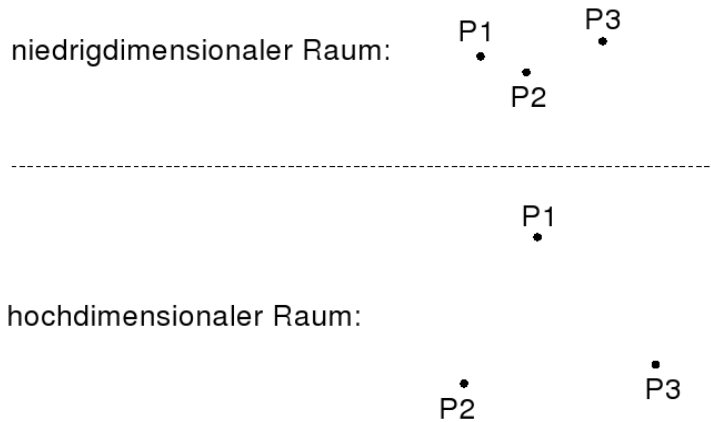
### Annäherung der Distanzen im hochdimensionalen Raum

steigender Erwartungswert und konstante Standardabweichung führen zu:

$$\lim_{d \rightarrow \infty} \frac{d_{max}}{\mu} \rightarrow 1 \quad \text{und} \quad \lim_{d \rightarrow \infty} \frac{d_{min}}{\mu} \rightarrow 1$$

Distanzen nähern sich also einander an  
→ Clustering ergibt aufgrund fehlender Lokalität wenig Sinn

### Annäherung der Distanzen in hochdim. Raum graphisch



Suchbäume clustern mehrere Feature-Objekte anhand einer geometrischen Figur (etwa MBR)

minimale Distanz zu eingeschlossenem Feature-Objekt darf nicht kleiner als zum Cluster sein

Approximationsfehler: durchschnittliche Differenz zwischen Distanz vom Anfragepunkt zum nächsten Feature-Objekt und zum dazugehörigen Cluster

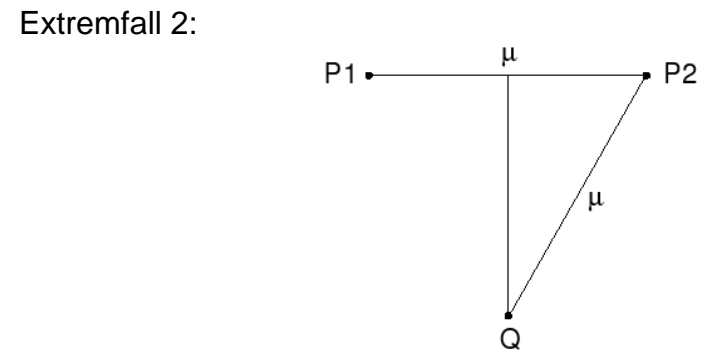
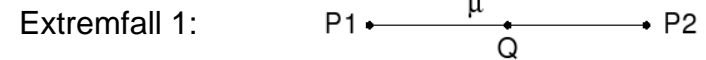
Ziel der Clusterung: Minimierung des Approximationsfehlers durch minimale geometrische Figur (etwa MBR)

Approximationsfehler steigt linear mit Erwartungswert der Distanzverteilung

Nachweis am minimalen konvexen Cluster: konvexe Hülle zwischen zwei Punkten  
→ also Linie

Extremfall: Anfragepunkt auf der Linie

Extremfall: Anfragepunkt als Punkt auf Simplex



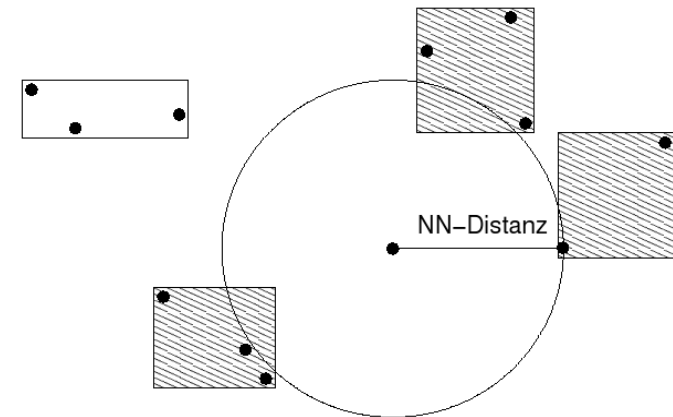
## Ineffiziente Suche in hochdimensionalen Indexbäumen

Ausschluss eines Teilbaums durch Vergleich von Clusterdistanz mit NN-Kandidatendistanz zum Anfragepunkt

optimaler (aber nicht realisierbarer) Suchalgorithmus:

- ♦ Annahme: Distanz zum nächsten Nachbar sei schon bekannt
- ♦ alle Cluster, deren Minimaldistanz kleiner als NN-Distanz ist, werden durchsucht

Optimaler Suchalgorithmus dient zur Abschätzung der minimal möglichen Suchkosten



## Optimaler Suchalgorithmus im hochdimensionalen Raum

wenn Dimensionszahl steigt:

- ♦ steigender Approximationsfehler und konstanter Abstand zwischen größter und kleinster Distanz
- ♦ Folge: selbst bei optimalem Suchalgorithmus können keine Cluster von Suche ausgeschlossen werden

## Optimaler Suchalgorithmus im hochdimensionalen Raum (2)

Fazit: ab bestimmter Dimensionsanzahl ist NN-Suche anhand euklidischer Distanzen mit Baumindizes nicht effizient (mindestens sequentieller Aufwand erforderlich)

Bemerkung: Problem gilt auch für Metrik-Bäume wie den M-Baum

