Umgang mit Räumen hoher Dimension in der Bildsuche

Vorlesung Bildverarbeitung 2, Teil 5
Wintersemester 2001/2002
Ullrich Köthe, FB Informatik, Uni Hamburg

„Course of Dimensionality“

- **N-dimensionale Räume ($N \gg 1$) verhalten sich anders als erwartet, z.B.:**
  - N-dimensionale Verallgemeinerungen von schnellen Suchverfahren sind langsamer als einfache lineare Suche
  - N-dimensionale Verteilungen haben überraschende Eigenschaften
  - Nächste-Nachbar Suche wird instabil, d.h. eine kleine Verschiebung des Anfrageobjekts kann zu völlig anderen Ergebnissen führen
- **studieren einige Eigenschaften N-dimensionaler Verteilungen**

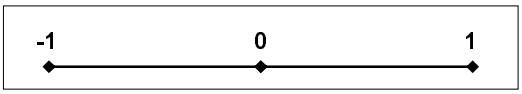
Gleichverteilung in der N-dimensionalen Kugel (1)

- **N=1: Intervall [-1,1]**
  - Wahrscheinlichkeit für Abstand vom Ursprung :

$$dP = P(r \in dr) = \frac{dr}{2}$$

  - Erwartungswert des Abstandes vom Ursprung:

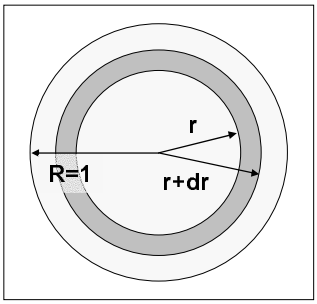
$$E[r] = \int_{-1}^{1} |r| \, dP = 2\int_{0}^{1} r \frac{dr}{2} = \frac{r^2}{2}\Big|_0^1 = \frac{1}{2}$$



Gleichverteilung in der N-dimensionalen Kugel (2)

- **N=2: Kreisfläche mit Radius 1**
  - Wahrscheinlichkeit für Abstand von Ursprung:

$$\begin{aligned} dP &= P(r_0 \in [r, r+dr]) \\ &= \pi(r+dr)^2 - \pi r^2 \\ &\approx 2\pi r \, dr \end{aligned}$$



  - Erwartungswert des Abstandes vom Ursprung:

$$E[r] = \int_{0}^{1} r \, dP = \int_{0}^{1} r \, 2\pi r \, dr = 2\frac{r^3}{3}\Big|_0^1 = \frac{2}{3}$$

Gleichverteilung in der N-dimensionalen Kugel (3)

- **Gleichverteilung in der Kugel für beliebige N**
  - Wahrscheinlichkeit für Abstand von Ursprung:

$$dP = P(r_0 \in [r, r+dr])$$

$$= \frac{V_N (r+dr)^N - V_N r^N}{V_N}, \qquad V_N \ldots \text{Volumen der N-dimensionalen Einheitskugel}$$

$$\approx N r^{N-1} dr$$

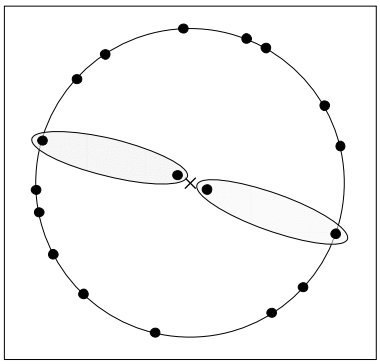
  - Erwartungswert des Abstandes vom Ursprung:

$$E[r] = \int_0^1 r\, dP = \int_0^1 N r^N dr = N \left. \frac{r^{N+1}}{N+1} \right|_0^1 = \frac{N}{N+1}$$

  ⇨ fast alle Punkte sind auf der Schale der N-dimensionalen Kugel

Gleichverteilung in der N-dimensionalen Kugel (4)

- **fast alle Punkte auf der Schale der N-dimensionalen Kugel ⇨ mögliche Instabilität der NN-Bestimmung:**
  - kleine Verschiebung des Anfragepunktes führt zu vollkommener Änderung der Region der nächsten Nachbarn



Gleichverteilung in der N-dimensionalen Kugel (5)

Abschätzung für nächsten-nachbar Abstand von *m* Punkten in N-d:
Durchmesser der Voronoiregionen

$$N V_N \approx m V_{N-1} \varepsilon^{N-1}, \quad N = 2k+1$$

$$V_{2k} = \frac{\pi^k}{k!}, \quad V_{2k+1} = \frac{2^{2k+1} k!\, \pi^k}{(2k+1)!}$$

$$(2k+1)\frac{2^{2k+1} k!\, \pi^k}{(2k+1)!} = m\frac{\pi^k}{k!}\varepsilon^{2k}$$

$$\varepsilon^{2k} = \frac{2^{2k+1}(k!)^2}{m(2k)!} \approx \frac{2^{2k+1}\left(\sqrt{2\pi k}\, k^k e^{-k}\right)^2}{m\sqrt{4\pi k}(2k)^{2k} e^{-2k}} = \frac{2\sqrt{\pi k}}{m}$$

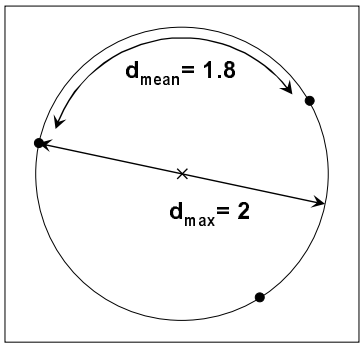
Approximation:
Voronoiregionen sind (N-1)-dimensionale Kugeln, die die Oberfläche vollständig bedecken

Gleichverteilung in der N-dimensionalen Kugel (6)

- **Beispiel: N=100, m=10^6:** $d = 2\varepsilon \approx 2\left(\frac{2\sqrt{\pi k}}{m}\right)^{\frac{1}{2k}} \approx 1.8$

  ⇨ **Schale immer noch dünn besetzt**

  „empty space problem“



N-dimensionale Gaußverteilung (1)

- **N-dimensionale Gaußverteilung:** $p(\vec{x}) = \frac{1}{(2\pi\sigma^2)^{N/2}} e^{-\frac{\vec{x}^2}{2\sigma^2}}$
- **Wahrscheinlichkeit für den Abstand vom Ursprung:**

$$p(r < R) = \frac{N V_N}{(2\pi\sigma^2)^{N/2}} \int_0^R r^{N-1} e^{-\frac{r^2}{2\sigma^2}} dr$$

	r = σ	r = 2σ	r = 3σ	r = 4σ	r = 8σ	r = 12σ	r = 16σ
N=1	0.68	0.95	0.997	0.9999	~1	~1	~1
N=2	0.39	0.86	0.989	0.9997	~1	~1	~1
N=100	~0	~0	~0	~0	0.002	0.997	~1

N-dimensionale Gaußverteilung (2)

- **Erwartungswert für das Abstandquadrat zweier Gaußverteilter Zufallszahlen (1D): Dipolmoment**

$$E\left[d(r_1, r_2)^2\right] = E\left[(r_1 - r_2)^2\right] = \frac{1}{2\pi\sigma^2} \iint (r_1 - r_2)^2 e^{-\frac{r_1^2 + r_2^2}{2\sigma^2}} dr_1\, dr_2$$

$$= 2\sigma^2$$

- **Varianz:**

$$Varianz\left[d(r_1, r_2)^2\right] = E\left[(r_1 - r_2)^4\right] - E\left[(r_1 - r_2)^2\right]^2$$

$$= \frac{1}{2\pi\sigma^2} \iint (r_1 - r_2)^4 e^{-\frac{r_1^2 + r_2^2}{2\sigma^2}} dr_1\, dr_2 - 4\sigma^4 = 8\sigma^4$$

N-dimensionale Gaußverteilung (3)

- **betrachte Vektoren von N Gaußverteilten Zahlen:**
- **Erwartungswert der Abstände:**

$$E\left[d_N(\vec{r}_1,\vec{r}_2)^2\right] = E\left[\sum_i (r_{i1}-r_{i2})^2\right] = \sum_i E\left[(r_{i1}-r_{i2})^2\right] = N2\sigma^2$$

- **Varianz der Abstände:**

$$Var\left[{d_N}^2\right] = Var\left[\sum_i (r_{i1}-r_{i2})^2\right] = \sum_i Var\left[(r_{i1}-r_{i2})^2\right] = N8\sigma^4$$

- **Verhältnis Standardabweichung zu Erwartungswert:**

$$\frac{\sqrt{Var\left[{d_N}^2\right]}}{E\left[{d_N}^2\right]} = \frac{\sqrt{2N}2\sigma^2}{2N\sigma^2} = \frac{\sqrt{2}}{\sqrt{N}} \begin{cases} \to 0 & (N\to\infty) \\ = 14\% & (N=100) \end{cases}$$

Verallgemeinerung für beliebige N-dimensionale Verteilungen (1)

- **Satz von Beyer et al.: falls**

$$\lim_{N\to\infty} \frac{\sqrt{Var\left[{d_N}^p\right]}}{E\left[{d_N}^p\right]} = \to 0 \quad (p>0)$$

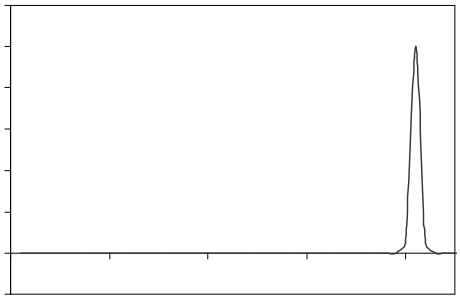
folgt:

$$\lim_{N\to\infty} P\left[\frac{\max(d_N)-\min(d_N)}{\min(d_N)} \le \varepsilon\right] = 1 \quad (\forall \varepsilon > 0)$$

$$\text{bzw.} \quad \frac{\max(d_N)-\min(d_N)}{\min(d_N)} \to_P 0$$

Verallgemeinerung für beliebige N-dimensionale Verteilungen (2)

- **Voraussetzung des Satzes von Beyer et al. gilt für viele Verteilungen, z.B. stets dann, wenn die Koordinaten unabhängig sind und identische Verteilungen haben (wie N-dimensionaler Gauß)**
- **typische Verteilung der Abstände vom Anfragepunkt (für große N):**

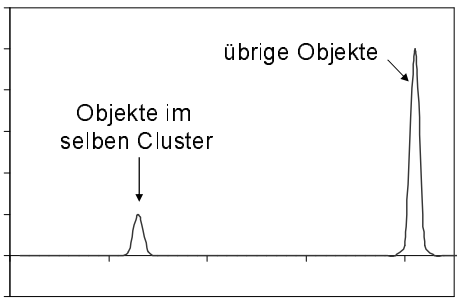


⇨ **Instabilität der NN-Suche ist in N-D sehr verbreitet (praktisch: ab ca. 10D)**

Wann ist NN-Suche stabil und sinnvoll? (1)

- **Verwendung anderer Abstände?**
  - alle L_p-Distanzen leiden unter dem Problem
  - Verhalten der *earth mover distance* unbekannt
- **Daten besitzen günstige Verteilung**
  - Daten bilden Cluster, und Anfrageobjekt liegt in einem Cluster

    beachte: abgesehen von der Bestimmung der Clusterzugehörigkeit kann NN-Abstand immer noch ohne Bedeutung sein



Wann ist NN-Suche stabil und sinnvoll? (2)

- **alle Achsen sind unabhängig und haben die gleiche Verteilung, aber Eigenwerte der Kovarianzmatrix nehmen schnell genug ab**
  - Beispiel: Gaußverteilungen mit $\sigma_i^2 = \sigma_1^2 / i^\lambda$

$$\mathrm{C}(d_N) = \frac{\sqrt{Var\left[{d_N}^2\right]}}{E\left[{d_N}^2\right]} = \frac{\sqrt{8\sigma_1^4 \sum_{i=1}^{N} 1/i^{2\lambda}}}{2\sigma_1^2 \sum_{i=1}^{N} 1/i^{\lambda}} \underset{N\to\infty}{\to} \begin{cases} 0 & (\lambda \le 1) \\ C_\lambda & (\lambda > 1) \end{cases}$$

$$\lambda = 1: \quad \sum_{i=1}^{N} \frac{1}{i^2} \to \frac{\pi^2}{6}, \quad \sum_{i=1}^{N} \frac{1}{i} \to \ln N, \quad \mathrm{C}(d_N) \sim \frac{\pi}{\sqrt{3}\ln N} \qquad (N = 100: \ 35\%)$$

$$\lambda = 2: \quad \sum_{i=1}^{N} \frac{1}{i^4} \to \frac{\pi^4}{90}, \quad \sum_{i=1}^{N} \frac{1}{i^2} \to \frac{\pi^2}{6}, \quad \mathrm{C}(d_N) = \frac{2}{\sqrt{5}} \qquad (89\%)$$

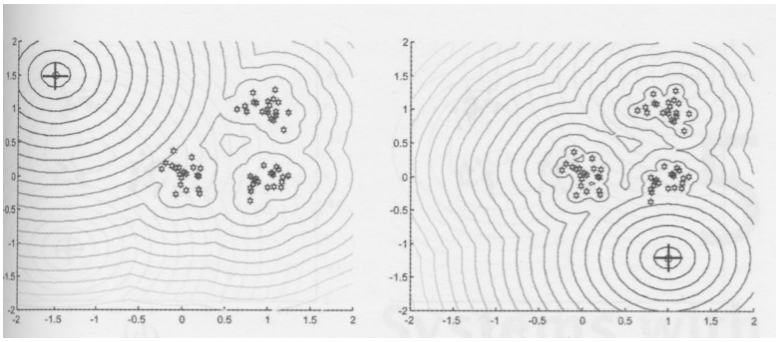
Wann ist NN-Suche stabil und sinnvoll? (3)

- **Verteilungen, deren intrinsische Dimension n≪N, z.B.:**
  - einige Achsen sind irrelevant (enthalten nur Rauschen)
  - Achsen sind korreliert, z.B. alle Daten liegen auf einer n-dimensionalen Hyperebene oder Hyperfläche
- **Modifikation des Merkmalsraums entsprechend der Aufgabenstellung**

  „Bedeutung eines Bildes bzw. seiner Merkmale hängt vom Kontext ab“
  - Gewichtung der Achsen
  - nichtlineare Verformung des Merkmalsraumes aufgrund der Lage des Anfragebildes oder eines Referenzbildes
  - Modifikation der Ähnlichkeitsfunktion
  - abgesehen von Relevanz-Feedback weitgehend unerforscht

Wann ist NN-Suche stabil und sinnvoll? (4)

- **nichtlineare Verzerrung der Merkmalsraums aufgrund der Lage des Anfrage-/Referenzbildes**



Wann ist NN-Suche stabil und sinnvoll? (5)

- **beides (Cluster und geringere intrinsische Dimension) kann gleichzeitig der Fall sein**
  - Verteilung der Clusterzentren mit n_0 Freiheitsgraden
  - Verteilung in jedem Cluster mit n_i Freiheitsgraden
  - hierarchische Clusterung möglich
  - Beispiel Fahrzeug-Klassifikation
    - Cluster Ebene 1: PKW, LKW, Motorrad etc.
    - Cluster Ebene 2: Motor, Kosten, Größe, Ausstattung,... mit Nutzer-abhängiger Gewichtung
    - unterschiedliche Dimensionen in jedem Cluster (Motor: Leistung, Verbrauch, Geschwindigkeit,...; Ausstattung: ...)

Dimensionsreduktion

Cluster-erhaltende Reduktion von N- auf n-D notwendig

- **lineare Verfahren:**
  - Hauptkomponentenanalyse (PCA) – Maximierung der Varianz
  - *Fisher´s Linear Discriminant* (FLD) – Maximierung der Unterscheidbarkeit der Cluster
  - *Relevant Component Analysis* (RCA) – Approximation von FDA ohne vollständige Klassifikation
  - *Projection Pursuit* – Maximierung anderer Clusterungsmaße
- **nicht-lineare Verfahren (Auswahl):**
  - *Multi-dimensional Scaling*: Distanzerhaltende Transformation
  - FastMap-Algorithmus (Approximation von MDS)
  - Neuronale Netze (z.B. Kohonen´s selbstorganisierende Karte)

Lineare Dimensionsreduktion (1)

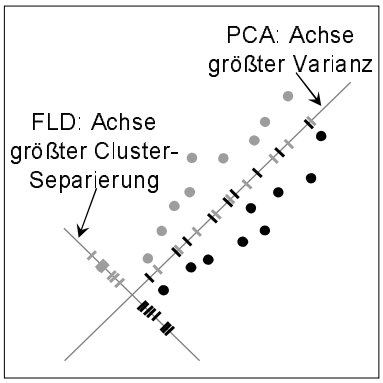
- **Suche die 1-dimensionale Projektion der Daten mit maximaler Varianz:**
  - suche a, so daß Var(x´) = Var(a^Tx) maximal
  - Rekursion: wiederhole dies n-mal in den Unterräumen, die orthogonal zu den bereits festgelegten Projektionen sind
- **Äquivalent zur Hauptkomponentenanalyse:**
  - maximiere die Determinante der transformierten Kovarianzmatrix
  - verwende die zu den n größten Eigenwerten gehörenden EV

$$S = \sum_{i=1}^{m} (\vec{x}_i - \vec{\mu})(\vec{x}_i - \vec{\mu})^T \qquad \text{Kovarianzmatrix}$$

$$A_{opt} = \arg\max_{A} \left| A^T S\, A \right| = [a_1\, a_2 \ldots a_m] \qquad \text{Eigenvektormatrix}$$

Lineare Dimensionsreduktion (2)

- **PCA berücksichtigt die Cluster nicht**
- ***Fisher´s linear discriminant* maximiert Varianz zwischen den Clustern und minimiert Varianz in den Clustern**



m_i – Größe Cluster i

$m = \sum_i m_i$ – Gesamtzahl der Punkte

$\vec{\mu}_i = \frac{1}{m_i} \sum_{\vec{x}_k \in C_i} \vec{x}_k$ – Zentrum Cluster i

$\vec{\mu} = \frac{1}{m} \sum_k \vec{x}_k$ – Schwerpunkt der Daten

Lineare Dimensionsreduktion (3)

- **Beschreibe Variabilität innerhalb der Klassen und zwischen den Klassen**

$S_C = \sum_{i=1}^{C} |m_i| (\vec{\mu}_i - \vec{\mu})(\vec{\mu}_i - \vec{\mu})^T$ between-class Kovarianz

$S_W = \sum_{i=1}^{C} \sum_{x_k \in m_i} (\vec{x}_k - \vec{\mu}_i)(\vec{x}_k - \vec{\mu}_i)^T$ Summe in-class Kovarianzen

- **maximiere die Diskrimination:**

$$A_{opt} = \arg\max_A \frac{|A^T S_C A|}{|A^T S_W A|} = [a_1 a_2 \ldots a_{c-1}] \quad \text{mit}$$

$S_C a_i = \lambda_i S_W a_i$ verallgemeinertes Eigenwertproblem

- **maximal (*c* – 1) verschiedene Eigenwerte**

Lineare Dimensionsreduktion (4)

- **Clusterzugehörigkeit ist oft nicht bekannt, schwer / mühsam zu bekommen, nicht allgemein zu definieren**
- **aber partielle Equivalenz ist oft vorhanden, z.B. zusammengehörende Bilder einer Serie / eines Shots ⇨ „*chunklets*“**
- **nutze *chunklets*, um irrelevante Achsen zu eliminieren / relevante zu finden ⇨ „*relevant components analysis*“**
- **Approximation der Diskriminantenanalyse ohne vollständige Klassifikation**

Lineare Dimensionsreduktion (5)

- **Kovarianzmatrizen wie FLD (chunklets statt Cluster)**
  1. bestimme effektiven Rang *r* (Anzahl der EW $\neq 0$) von S_C
  2. führe PCA mit S_W durch und reduziere Daten auf *r* Dimensionen
  3. berechne S_C im reduzierten (*r*-dimensionalen) Raum und berechne Hauptkomponenten von $S_{C\text{-}r}$
  4. reduziere Dimension weiter entsprechend der Hauptkomponenten von $S_{C\text{-}r}$
- **gute Approximation von FLD, falls die Mittelwerte der chunklets die Mittelwerte der entsprechenden Cluster approximieren**
- **Test mit einer Gesichtsdatenbank: deutlich bessere Ergebnisse als Eigenfaces, fast so gut wie Fisherfaces**

Lineare Dimensionsreduktion (6)

- **falls keine Cluster oder chunklets verfügbar: suche Projektion, die (anstelle der Varianz) Maß für die Clusterung der Verteilung maximiert**
  ⇨ *„projection pursuit“*-Methode
- **geeignete Maße: z.B. aus Kumulanten der Verteilung:**

$k_3 = \mu_3 - 3\mu_1\mu_2 + 2\mu_1^3$ Skewness (Asymmetrie) der Verteilung

$k_4 = \mu_4 - 4\mu_3\mu_1 - 3\mu_2^2 + 12\mu_1^2\mu_2 - 6\mu_1^4$ Kurtosis (Exzentrizität)

mit $\mu_i = E[x^i]$ – Momente der Verteilung

berechne $k_3 / k_2^{3/2}$ oder k_4 / k_2^2 oder $k_3^2 + \frac{1}{4}k_4^2$ (=0 ohne Cluster)

Lineare Dimensionsreduktion (7)

- ***„projection pursuit“*-Algorithmus:**
  1. bestimme a_1 (erste Hauptkompenentenrichtung)
  2. projiziere auf a_1 und bestimme Maß der Clusterung
  3. verbessere a_1 durch Gradientenabstieg (keine geschlossene Lösung bekannt) und gehe zu 2. bis sich nichts mehr ändert
  4. wiederhole den Algorithmus rekursiv im zu a_1 senkrechten Raum
- **Elimination von Ausreißern notwendig**
- **Bewertung: noch keine Verwendung bei der Bildsuche bekannt**

Nichtlineare Dimensionsreduktion: *multidimensional scaling* (1)

- **Cluster bleiben automatisch erhalten, wenn bei der Dimensionsreduktion die Distanzmatrix erhalten bleibt**

$$\delta_{ij} = \delta(x_i, x_j) \quad x_i, x_j : \text{ Datenpunkte}$$

- **δ() kann beliebige Abstandsfunktion sein; die Punkte müssen nicht in einem Vektorraum liegen**
- **definiere Maß der Veränderung der Abstände durch Projektion $x_i \rightarrow p_i \in \mathbb{R}^n$: „*multidimensional scaling*“**

$$\text{minimiere} \quad \text{Stress}(d,\delta) = \sum_{i,j} \left(\|p_i - p_j\| - \delta_{ij} \right)^2$$

- **nichtlineare Projektion hat oft geringeren Stress()**

Nichtlineare Dimensionsreduktion: *multidimensional scaling* (2)

- **Minimierung von Stress() ist kompliziert:**
  ⇨ Vereinfachung

$$\text{minimiere} \quad \text{SStress}(d,\delta) = \sum_{i,j} \left(\|p_i - p_j\|^2 - \delta_{ij}^2 \right)^2$$

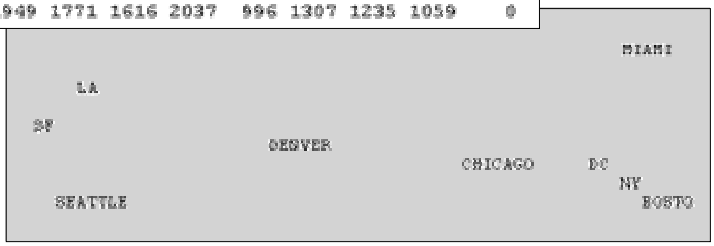
$$\text{minimiere} \quad \text{Strain}(d,\delta) = \sum_{i,j} \left(\|p_i - p_j\|^2 - \tau(\delta_{ij}^2) \right)^2$$

$$\text{mit} \quad \tau(\delta_{ij}^2) = -\frac{1}{2}\left(\delta_{ij}^2 - \overline{\delta_{i.}^2} - \overline{\delta_{.j}^2} + \overline{\delta_{..}^2} \right) \quad \text{Zentrierungstransf.}$$

- **Strain() hat geschlossene Lösung (EV von $\tau(\delta_{ij}^2)$), die als Startlösung für iterative Lösung von Stress() oder SStress() benutzt werden kann**
- **Lösung: Gradientenabstieg, Newton, neuronale Netze**

Nichtlineare Dimensionsreduktion: *multidimensional scaling* (3)

		1 BOST	2 NY	3 DC	4 MIAM	5 CHIC	6 SEAT	7 SF	8 LA	9 DENV
1	BOSTON	0	206	429	1504	963	2976	3095	2979	1949
2	NY	206	0	233	1308	802	2815	2934	2786	1771
3	DC	429	233	0	1075	671	2684	2799	2631	1616
4	MIAMI	1504	1308	1075	0	1329	3273	3053	2687	2037
5	CHICAGO	963	802	671	1329	0	2013	2142	2054	996
6	SEATTLE	2976	2815	2684	3273	2013	0	808	1131	1307
7	SF	3095	2934	2799	3053	2142	808	0	379	1235
8	LA	2979	2786	2631	2687	2054	1131	379	0	1059
9	DENVER	1949	1771	1616	2037	996	1307	1235	1059	0



Nichtlineare Dimensionsreduktion: *multidimensional scaling* (4)

- **aufwendiges Verfahren**
- **sehr gute Ergebnisse, wenn nur Distanzen bekannt sind (z.B. Chemie, Psychologie)**
- **in leicht modifizierter Form auch anwendbar wenn nur Rankings bekannt**
- **für die Bildsuche unbrauchbar, da es keine effiziente Möglichkeit gibt, die n-dimensionale Koordinate des Anfragebildes zu bestimmen**
  - Wiederholung der Optimierung nötig
  - Näherungsalgorithmen haben immer noch lineare Komplexität (Vergleich mit jedem Datenbankbild)

Nichtlineare Dimensionsreduktion: Pivot-basierte Verfahren (1)

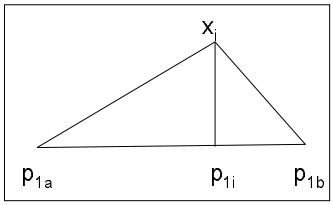
- **wähle n Pivotpunkte (Clusterzentren!) und verwende Vektor der Abstände als Punktkoordinate**
  - für alle Pivots gilt: $d(x_1, x_2) \geq |d(x_1,p) - d(x_2,p)|$ ⇨ schnelle Abschätzung des Abstandes $d(x_1, x_2)$
  - Verbesserung: Lipschitz-Einbettung, SparseMap-Algorithmus
- **FastMap-Algorithmus (Faloutsos & Lin ´95):**
  - rekursive Auswahl von Pivots
  - sehr schnell, aber relativ ungenau
  - für Bildrepräsentation nicht geeignet, aber für Optimierung der Ergebnispräsentation einer Suche (*star field display*, siehe später)

Nichtlineare Dimensionsreduktion: Pivot-basierte Verfahren (2)

- **FastMap-Algorithmus:**
  1. Auswahl eines Paars von Pivot-Punkten
     - wähle beliebigen Punkt $x_1´$ und finde den fernsten Punkt $x_2´$
     - wiederhole dasselbe für $x_2´$, und dann noch 3-mal
     - das letzte Paar bildet das Pivot-Paar (p_{1a}, p_{1b})
  2. projiziere alle übrigen Punkte x_i auf die Achse (p_{1a}, p_{1b})
     ⇨ Koordinate p_{1i} bzgl. erster Achse

$$d(x_i, p_{1a})^2 + p_{1i}^2 = d(x_i, p_{1b})^2 - \left(d(p_{1a}, p_{1b}) - p_{1i}\right)^2$$

$$p_{1i} = \frac{d(x_i, p_{1a})^2 + d(p_{1a}, p_{1b})^2 - d(x_i, p_{1b})^2}{2d(p_{1a}, p_{1b})}$$



Nichtlineare Dimensionsreduktion: Pivot-basierte Verfahren (3)

- **FastMap-Algorithmus (Fortsetzung):**
  3. berechne den „Restabstand“ nach Abzug von p_{1i}

$$d_1(x_i, x_j)^2 = d(x_i, x_j)^2 - \left(p_{1i} - p_{1j}\right)^2$$

  4. Wiederhole das Verfahren mit $d_1()$ und neuen Pivots (p_{2a}, p_{2b}) um die zweite Koordinate p_{2i} zu bestimmen usw. bis p_{ni}
- **Verfahren ist exakt, wenn d() die Euklidische Distanz (im originalen N-dimensionalen Vektorraum) ist**
- **für andere Distanzen auch anwendbar, aber nicht so gute Resultate; Versagen, falls $d_k()^2$ negativ wird**
- **gut für schnelle Dimensionsreduktion für 2D Anzeige**