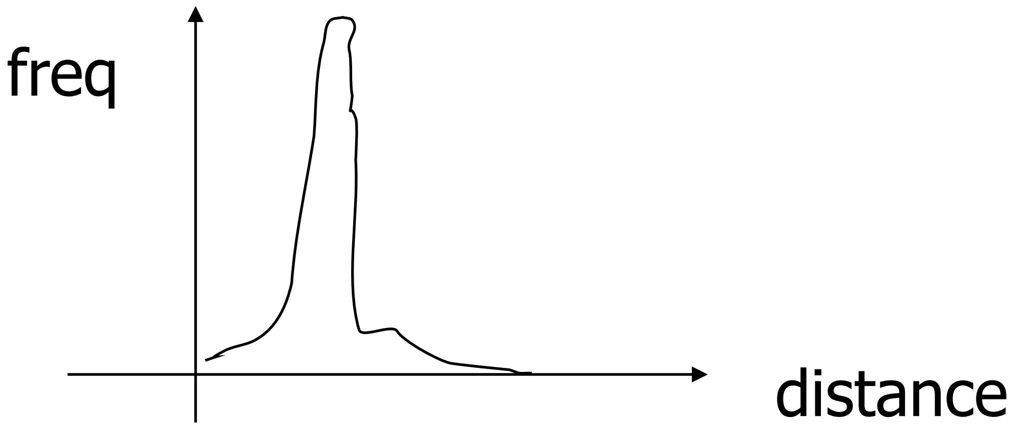
Dimensionality Reduction

Multimedia DBs

- Many multimedia applications require efficient indexing in high-dimensions (time-series, images and videos, etc)
- Answering similarity queries in high-dimensions is a difficult problem due to "curse of dimensionality"
- A solution is to use Dimensionality reduction

High-dimensional datasets

- Range queries have very small selectivity
- Surface is everything
- Partitioning the space is not so easy: 2^d cells if we divide each dimension once
- Pair-wise distances of points are very skewed



Dimensionality Reduction

- The main idea: reduce the dimensionality of the space.
- Project the d-dimensional points in a k-dimensional space so that:
  - k << d
  - distances are preserved as well as possible
- Solve the problem in low dimensions

Multi-Dimensional Scaling

- Map the items in a k-dimensional space trying to minimize the stress

$$stress = \sqrt{\frac{\sum_{i,j}(\hat{d}_{ij} - d_{ij})^2}{\sum_{i,j} d_{ij}^2}}, d_{ij} = | o_j - o_i | \quad and \quad \widehat{d}_{ij} = | \widehat{o}_j - \widehat{o}_i |$$

- Steepest Descent algorithm:
  - Start with an assignment
  - Minimize stress by moving points
- But the running time is $O(N^2)$ and $O(N)$ to add a new item

Embeddings

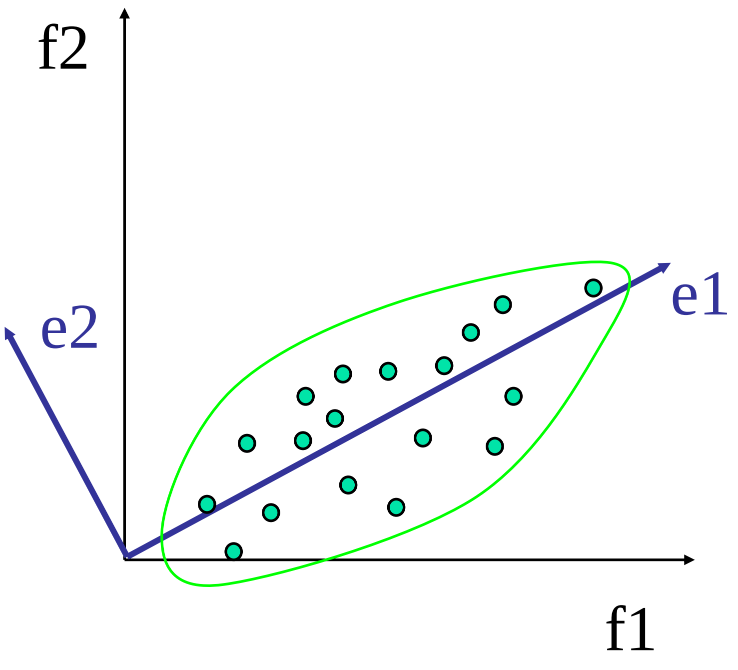
- Given a metric distance matrix D, embed the objects in a k-dimensional vector space using a mapping F such that
  - D(i,j) is close to D’(F(i),F(j))
- Isometric mapping:
  - exact preservation of distance
- Contractive mapping:
  - D’(F(i),F(j)) <= D(i,j)
- d’ is some Lp measure

GEMINI

- Using the contractive property (lower bounding lemma) we can show that we can use the index in the lower dimensional space to retrieve the exact answer for ε-range and NN query.
- GEMINI framework

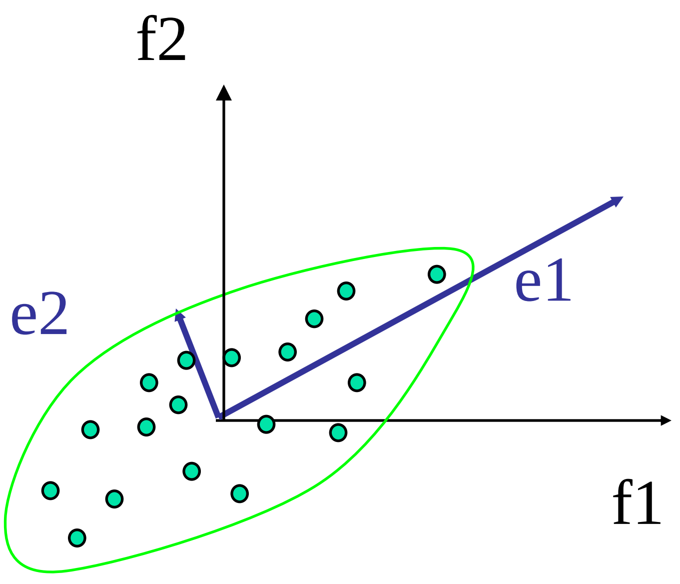
PCA

- Intuition: find the axis that shows the greatest variation, and project all points into this axis



SVD: The mathematical formulation

- Normalize the dataset by moving the origin to the center of the dataset
- Find the eigenvectors of the data (or covariance) matrix
- These define the new space
- Sort the eigenvalues in "goodness" order



SVD Cont'd

- Advantages:
  - Optimal dimensionality reduction (for linear projections)

- Disadvantages:
  - Computationally hard. … but can be improved with random sampling
  - Sensitive to outliers and non-linearities

SVD Extensions

- On-line approximation algorithm
  - [Ravi Kanth et al, 1998]
- Local dimensionality reduction:
  - Cluster the dataset, solve for each cluster
  - [Chakrabarti and Mehrotra, 2000], [Thomasian et al]

FastMap

What if we have a finite metric space (X, d)?

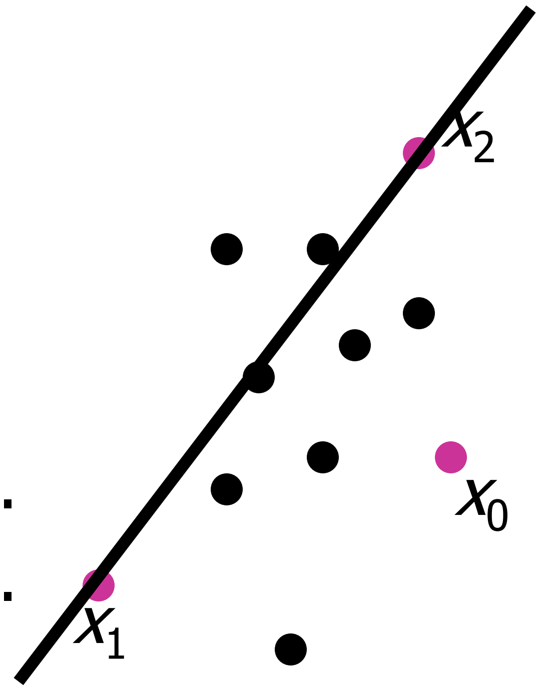
Faloutsos and Lin (1995) proposed FastMap as metric analogue to the KL-transform (PCA). Imagine that the points are in a Euclidean space.

- Select two **pivot points** x_a and x_b that are far apart.
- Compute a **pseudo-**projection of the remaining points along the "line" $x_a x_b$.
- **"Project"** the points to an orthogonal subspace and **recurse**.

Selecting the Pivot Points

The pivot points should lie along the principal axes, and hence should be far apart.

- Select any point x_0.
- Let x_1 be the furthest from x_0.
- Let x_2 be the furthest from x_1.
- Return (x_1, x_2).



Pseudo-Projections

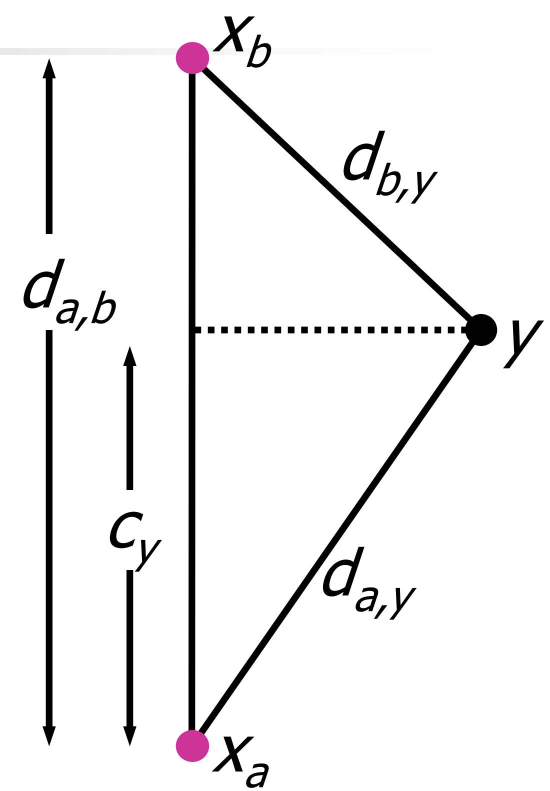
Given pivots (x_a , x_b), for any third point y, we use the **law of cosines** to determine the relation of y along $x_a x_b$.

$$d_{by}^2 = d_{ay}^2 + d_{ab}^2 - 2c_y d_{ab}$$

The **pseudo-projection** for y is

$$c_y = \frac{d_{ay}^2 + d_{ab}^2 - d_{by}^2}{2d_{ab}}$$

This is first coordinate.

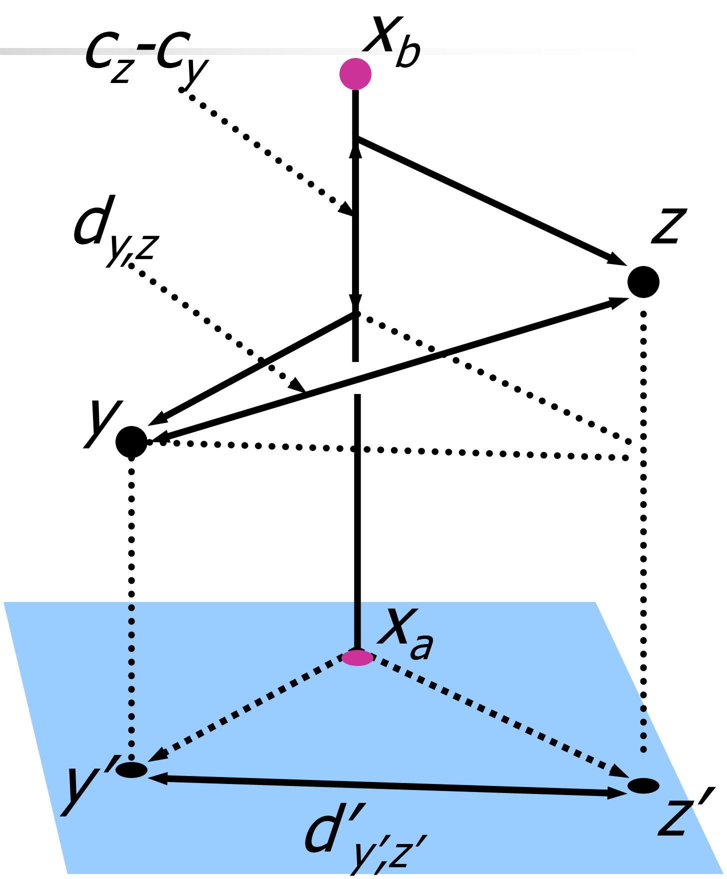


“Project to orthogonal plane”

Given distances along $x_a x_b$ we can compute distances within the “orthogonal hyperplane” using the Pythagorean theorem.

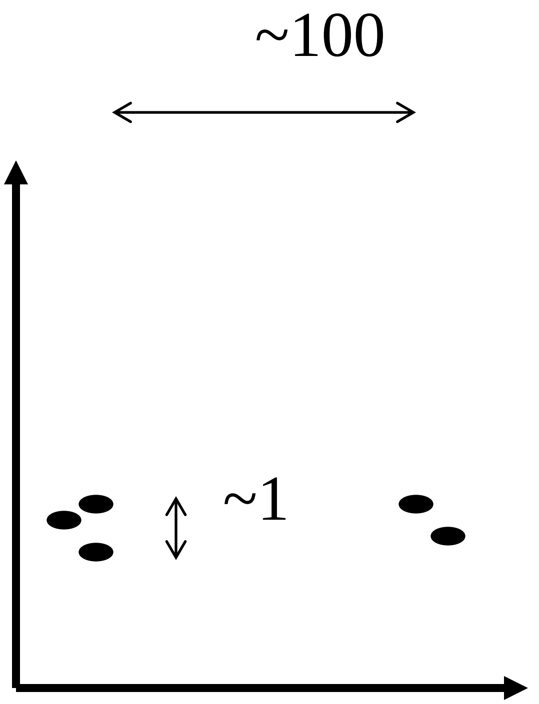
$$d'(y',z') = \sqrt{d^2(y,z) - (c_z - c_y)^2}$$

Using $d'(.,.)$, recurse until k features chosen.



Example

	O1	O2	O3	O4	O5
O1	0	1	1	100	100
O2	1	0	1	100	100
O3	1	1	0	100	100
O4	100	100	100	0	1
O5	100	100	100	1	0

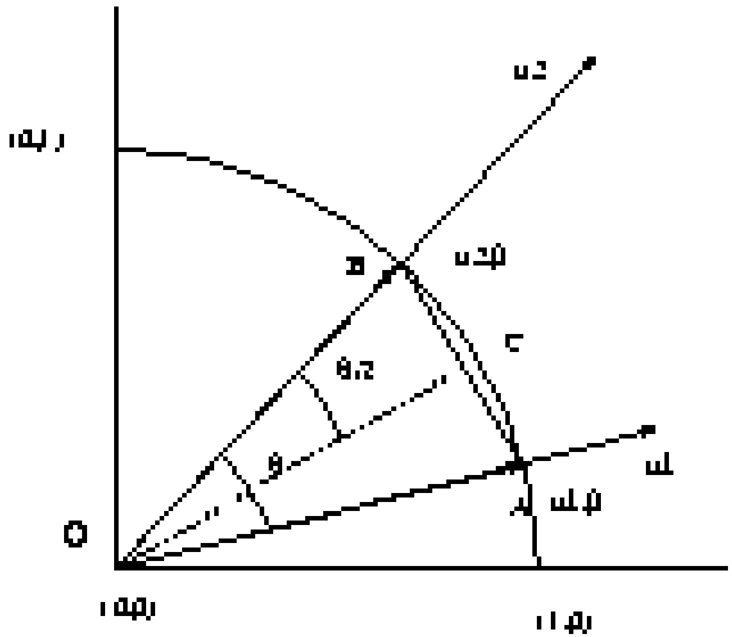


Example

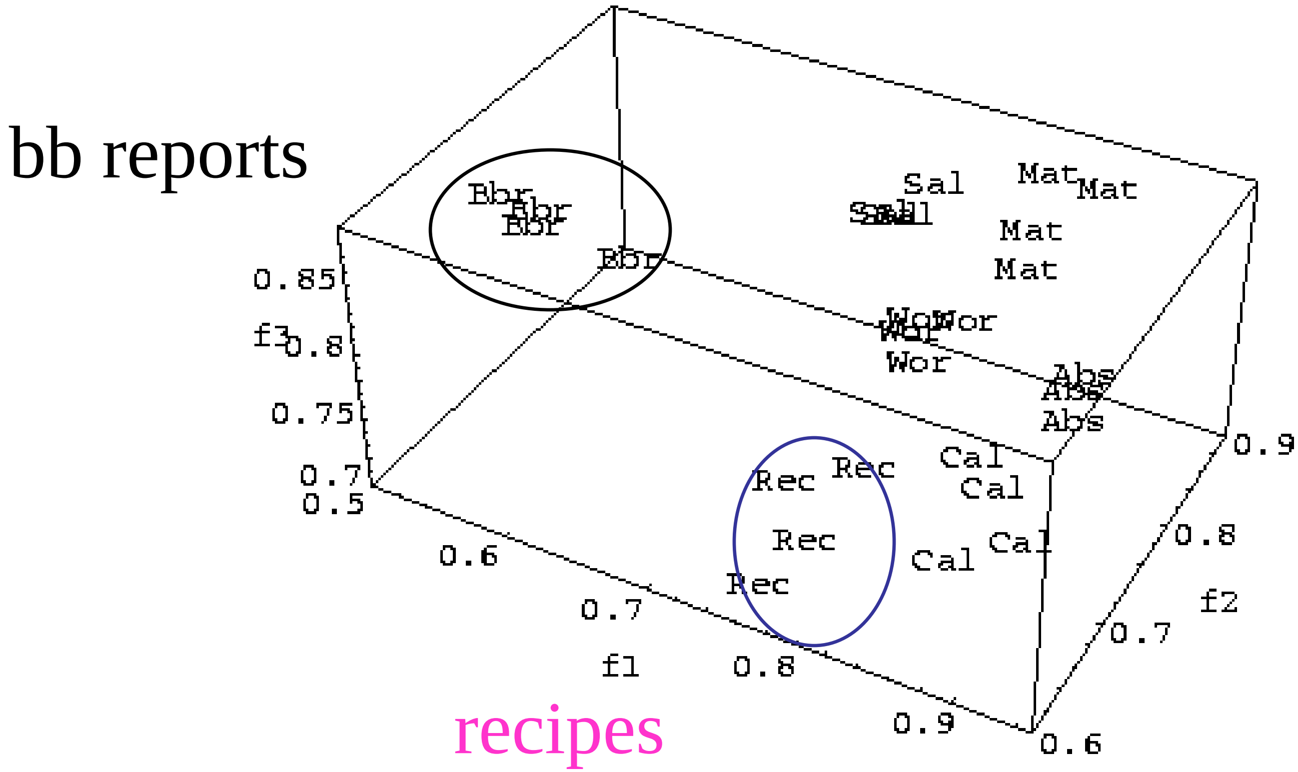
- Pivot Objects: O1 and O4
- X1: O1:0, O2:0.005, O3:0.005, O4:100, O5:99
- For the second iteration pivots are: O2 and O5

Results

Documents /cosine similarity -> Euclidean distance (how?)



Results



bb reports
Bbr
Sal
Mat
Wor
Abs
Cal
Rec
0.85
f3
0.8
0.75
0.7
0.5
0.6
0.7
f1
0.8
0.9
0.6
0.7
0.8
f2
0.9
recipes

FastMap Extensions

- If the original space is not a Euclidean space, then we may have a problem:

The projected distance may be a complex number!

- A solution to that problem is to define:

$$d_i(a,b) = \mathrm{sign}(d_i(a,b))\,(|\, d_i(a,b)\,|^2)^{1/2}$$

where, $d_i(a,b) = d_{i-1}(a,b)^2 – (x^i_a - x_b^i)^2$

Random Projections

- Based on the Johnson-Lindenstrauss lemma:
- For:
  - $0 < \varepsilon < 1/2$,
  - any (sufficiently large) set $\boldsymbol{S}$ of M points in R_n
  - $k = O(\varepsilon^{-2} \ln M)$
- There exists a linear map $f: \boldsymbol{S} \rightarrow R_k$, such that
  - $(1-\varepsilon) D(S,T) < D(f(S),f(T)) < (1+\varepsilon) D(S,T)$ for S,T in $\boldsymbol{S}$
- Random projection is good with constant probability

Random Projection: Application

- Set k = $O(\varepsilon^{-2} \ln M)$
- Select k random n-dimensional vectors
  - (an approach is to select k gaussian distributed vectors with variance 0 and mean value 1: N(1,0))
- Project the original points into the k vectors.
- The resulting k-dimensional space approximately preserves the distances with high probability
- Monte-Carlo algorithm: we do not know if correct

Random Projection

- A very useful technique,
- Especially when used in conjunction with another technique (for example SVD)
- Use Random projection to reduce the dimensionality from thousands to hundred, then apply SVD to reduce dimensionality farther