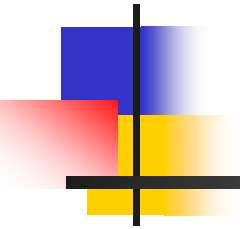
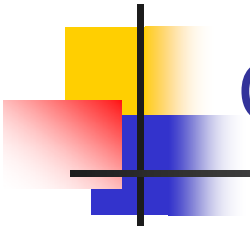


FastMap

C. Faloutsos and K.-I. Lin, “FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets”, *Proc. ACM SIGMOD*, 1995, 163-174



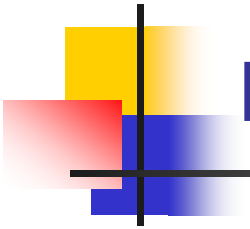
Outline

- Dimensionality reduction
- Existing method-MDS
- FastMap algorithm
- Experiments and Results
- Conclusion

Retrieval techniques for high-dimensional datasets

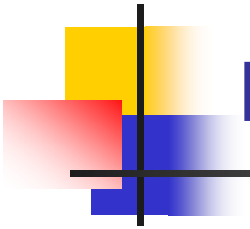
- The retrieval problem:
  - Given a set of objects *S*, and a query object q,

    find the objects that are most similar to q.
- Applications:
  - financial, voice, marketing, medicine
- Indexing for efficient and fast retrieval



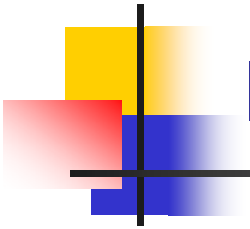
Disadvantages of high dimensions

- High-dimensionality:
  - decreases index structure performance (the curse of dimensionality)
  - slows down the distance computation
- Inefficiency-size of dataset is very large



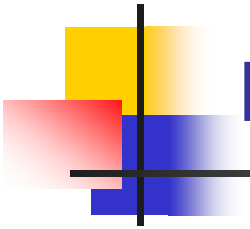
Dimensionality reduction

- The main idea: reduce the dimensionality of the space.
- Feature extraction
- Project the n-dimensional tuples that represent object in a k-dimensional space so that:
  - k << n
  - distances are preserved as much as possible
- Use indexing mechanism on reduced dimension for efficient retrieval



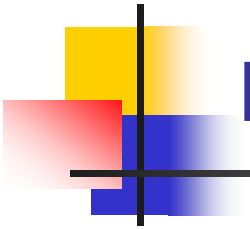
In lower dimensions...

- Indexing structures work much better in lower dimensionality spaces
- The distance computations run faster
- The size of the dataset is reduced, improving performance



Definitions

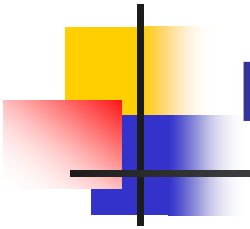
- K-dimensional point P_i that corresponds to object O_i is ‘image’ of O_i.
- $P_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,k})$
- K-dimensional space containing the ‘images’ is target space



Desirable Queries

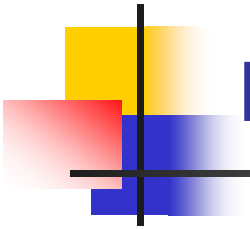
- *query-by-example* search a collection of objects to find the ones that are within a user-defined distance from the query object

- *all pairs query* find the pairs of objects which are within distance from each other



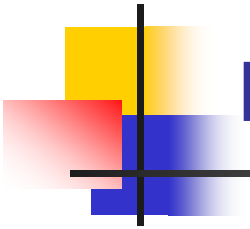
Problem Definition

- ‘Distance’ case (feature extraction):
  - Given N objects, N x N distance matrix
  - Find N points in k-dimensional space
- ‘Features’ case (dimensionality reduction):
  - Given N vectors with n attributes each
  - Find N vectors in k-dimensional space



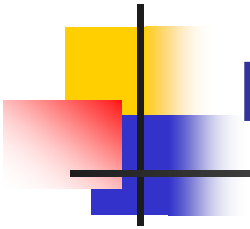
Mapping Requirements

- Fast to compute: O(*N*) or O(*N* log*N)*, but not O(N^2)
- Preserve distances with little discrepancies
- Should be very fast to map a new object
  - Required for answering ‘query-by-example’



Existing methods

- Number of dimensionality reduction techniques exist
  - DFT
  - SVD
  - Piecewise approximation
  - Multidimensional scaling
  - ...
- Most of these solve ‘features’ case



Multi-Dimensional Scaling

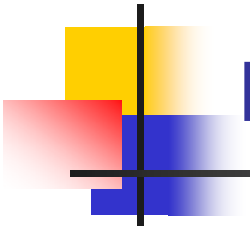
- Used to discover the underlying structure of a set of items, from the distances between them.
- Finds an embedding in k-dimensional Euclidean that minimizes the difference in distances.
- Has been applied to clustering, visualization, information retrieval...

Algorithm for MDS

- Input: N objects, their pairwise distances, the desired dimensionality k.
- Optimization criterion:

$$stress = \sqrt{\frac{\sum_{i,j}\left(\hat{d}_{ij} - d_{ij}\right)^2}{\sum_{i,j} d_{ij}^2}}$$

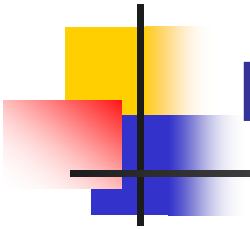
  where d_{ij} be the distance between objects O_i, O_j, and d_{ij}' be the Euclidean distance of the k-dim representations
- Basic algorithm:
  - start with an assignment (random or heuristic based)
  - minimize stress by moving points



Drawbacks of MDS

- Requires $O(N^2)$ time, which is impractical for large databases
- Fast retrieval is questionable as MDS is not prepared for “query-by-example” operation.

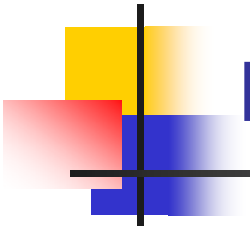
  Mapping a query is O(N) !



FastMap-Main Idea

- Given objects and distances, pretend objects are indeed in some n-d space
- Try to project these points on k mutually orthogonal directions
- But we have only distance matrix
- Assumption:
  - Distance is a metric i.e

    Reflexive, symmetric, obeying triangular inequality

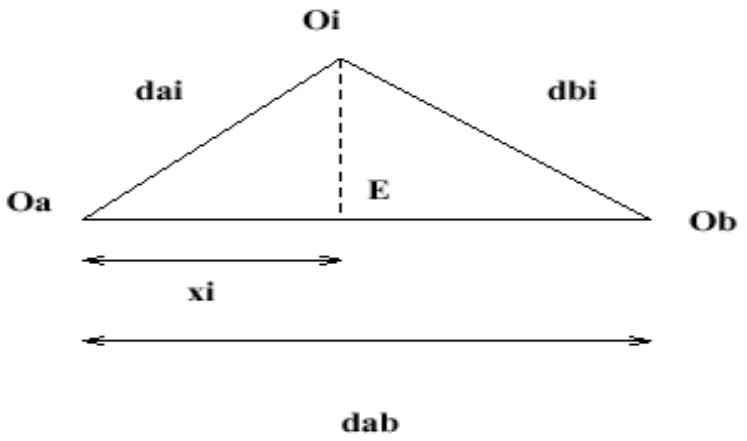


How FastMap Works

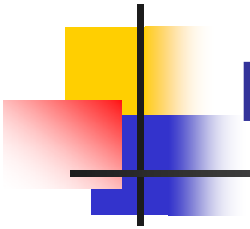
- Find two objects that are far away
- Project all points on the line the two objects define, to get the first coordinate
- Project all objects on a hyperplane perpendicular to the line the two objects define
- Repeat k-1 times

Method

- Can map points to a line segment, preserving some of the distance information
- Problem solved for k=1!

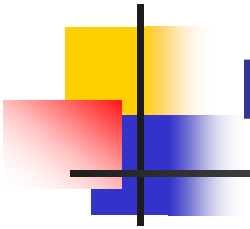


$$x_i = \frac{d_{a,i}^2 + d_{a,b}^2 - d_{b,i}^2}{2 d_{a,b}}$$



Extending the method

- In higher dimensions
  - Suppose the points are in n-dimensional space
  - Consider an (n-1)-dimensional hyper-plane, H, that is perpendicular to the line (O_a, O_b)
  - Project these objects on this hyper-plane
    - Let O_i' denote the projection of O_i
  - Then, the problem is the same as the original problem, with n and k decreased by 1



But...

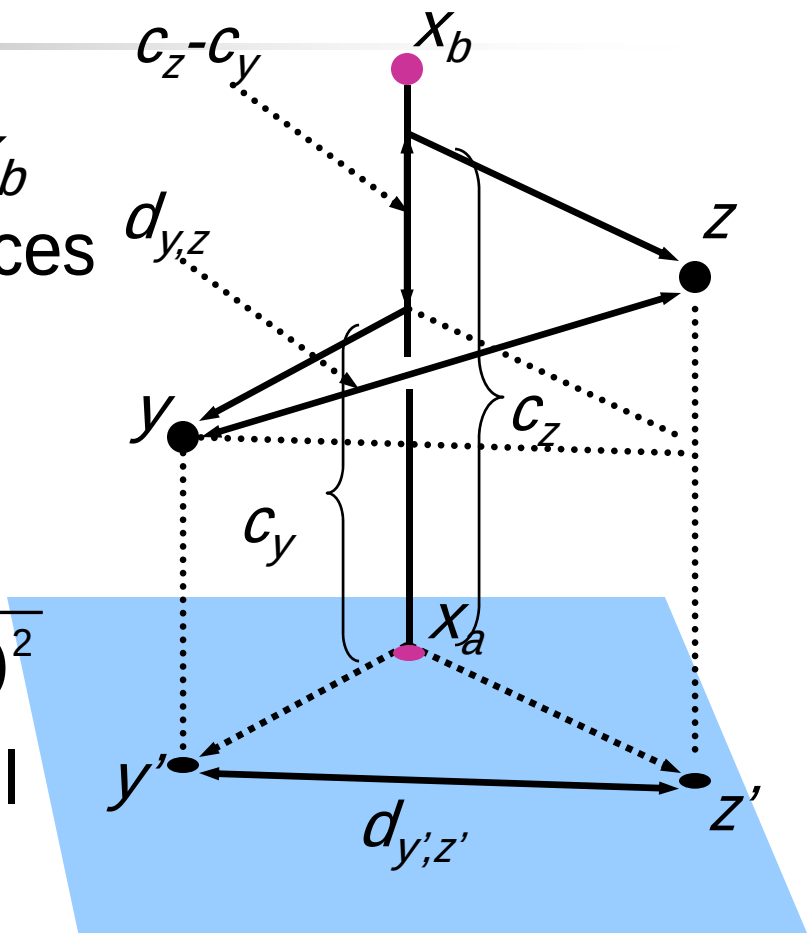
- The distance function D’() between two of the projections on hyper-plane H needs to be determined
- If D’() can be obtained in terms of D() then we can proceed recursively

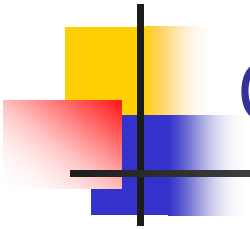
Project to orthogonal plane

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.





Choosing pivot points

- Choose O_a and O_b such that $D(O_a,O_b)$ is maximum
- Requires $O(N^2)$ distance computation
- Use heuristic to determine O_a and O_b

Heuristic for pivot points

Algorithm 1 *choose-distant-objects* ($\mathcal{O}$, *dist*())

begin

1) Chose arbitrarily an object, and declare it to be the second pivot object O_b
2) set O_a = (the object that is farthest apart from O_b) (according to the distance function *dist*())
3) set O_b = (the object that is farthest apart from O_a)
4) report the objects O_a and O_b as the desired pair of objects.

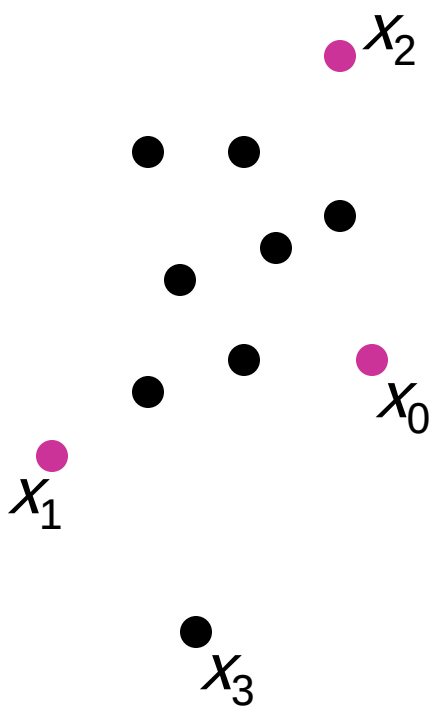
end

Step 2 and 3 can be repeated constant number of times

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 farthest from x_0.
- Let x_2 be the farthest from x_1.
- Return (x_1, x_2).



Algorithm

Global variables

- N × k array X[ ]
  - At algorithm's end, the i^{th} row of this array will be the image of the i^{th} point
- 2 × k pivot array PA[ ]
  - This stores the id's of the pivot points - one pair per recursive call
- col# = 0

Algorithm FastMap(k, D( ), O)

1. if ($k \leq 0$)
   return
   else
   col# ++
2. let O_a and O_b be the result of *Choose-Distant-Points*(O, D())

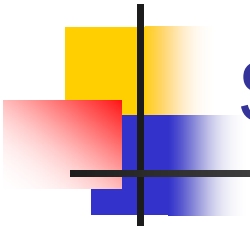
Algorithm contd.

3. PA[1, col#] = a ;PA[2, col#] = b
4. if $D(O_a, O_b) = 0$

   set X[i, col#] = 0 for every i and return
5. for each point O_i

   compute x_i using the equation and update the global array: X[i, col#] = x_i
6. New distance function D′( ) between projections is passed for recursion

   call FastMap(k-1, D′( ), O)



Some observations

- Complexity of FastMap is O(Nk)
- Pivot array PA[ ] stores pivot objects selected in each iteration
  - Useful to map a new query object
  - Complexity of mapping new query object independent of database size

Example

- Consider the following distance matrix

	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

O1,O2,O3 form one cluster O4,O5 form another

Example

- These are the results

$\mathbf{X}[]$	f_1	f_2	f_3
O1	0	0.707089	0.668149
O2	0.005	1.41418	0.935411
O3	0.005	1.06062	0
O4	100	0.707089	0.668149
O5	99.995	0	1

Example

- We have the *stress* function as,

$$stress = \sqrt{\frac{\sum_{i,j}\left(\hat{d}_{ij} - d_{ij}\right)^2}{\sum_{i,j} d_{ij}^2}}$$

- We would like to stress to be minimum

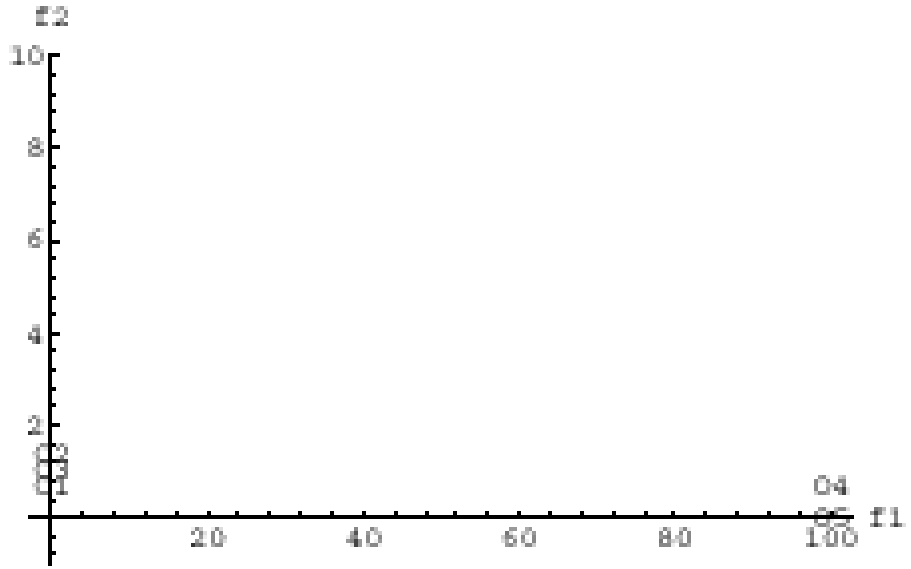
Example

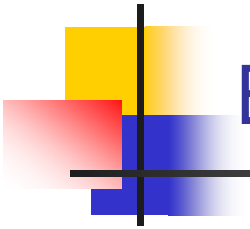
- The *stress* is,

Iteration#	Pivot	Stress
1	O1, O4	0.008
2	O5, O2	0.004
3	O3, O5	0.001

Example

- Using the first two coordinates, f_1 and f_2, of the target space, we plot the points



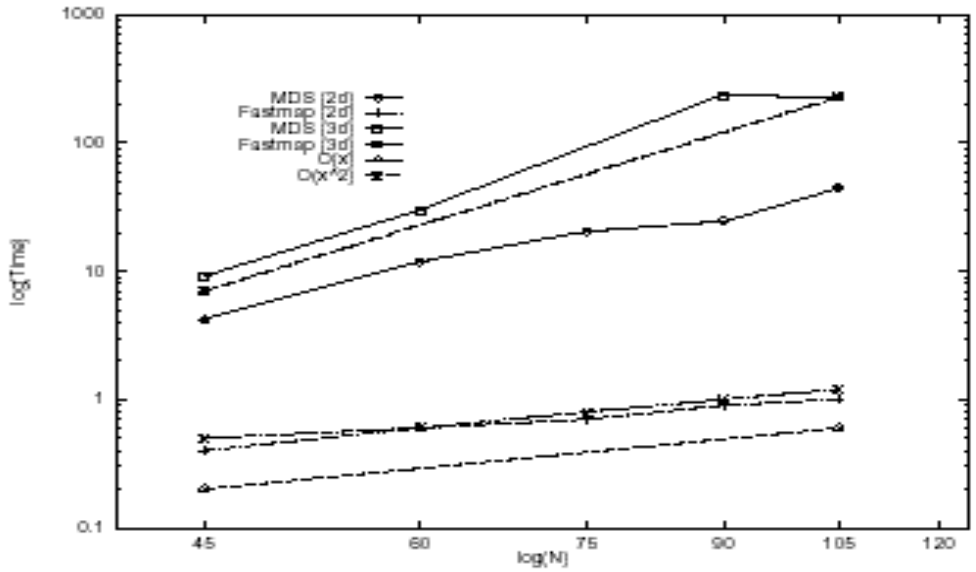


Experiments

- Compare FastMap with MDS
  - speed and quality
- Illustrate the visualization and clustering abilities
  - real and synthetic datasets
  - WINE dataset-result of chemical analysis of 3 varieties of wine grown in Italy
  - GAUSSIAN5D-120 points in 5-dimensional space with 6 clusters

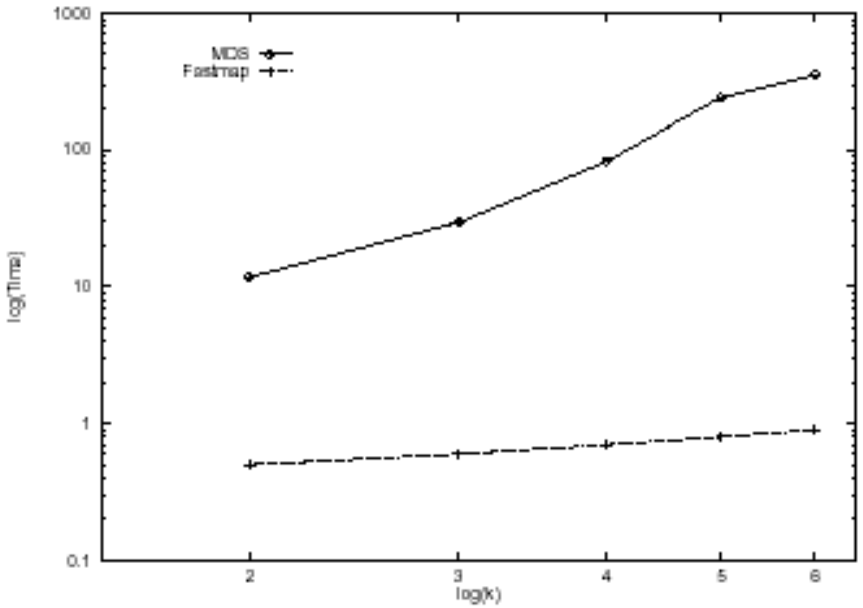
Comparison with MDS

- Response time vs. no. of database size



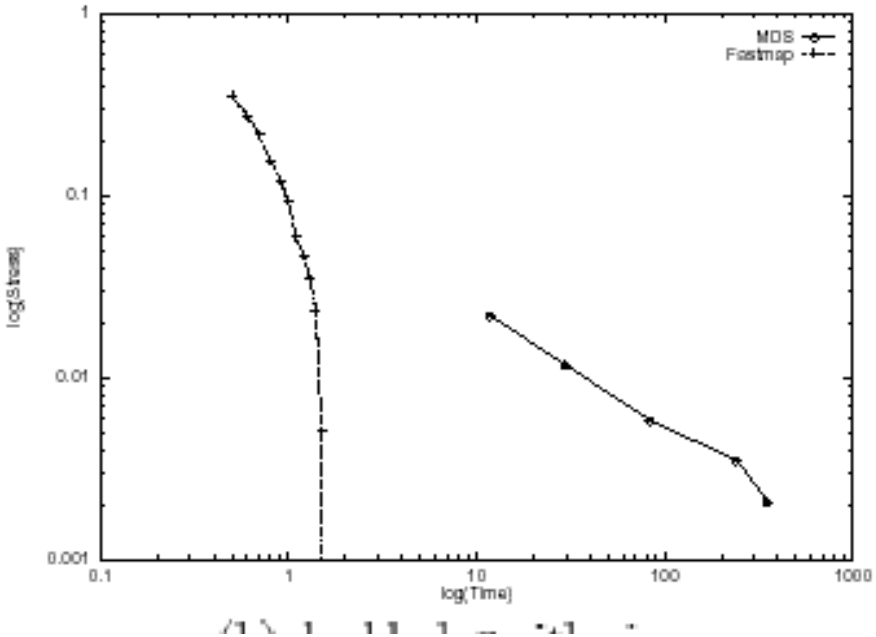
Comparison with MDS

- Response time vs. no. of dimensions k

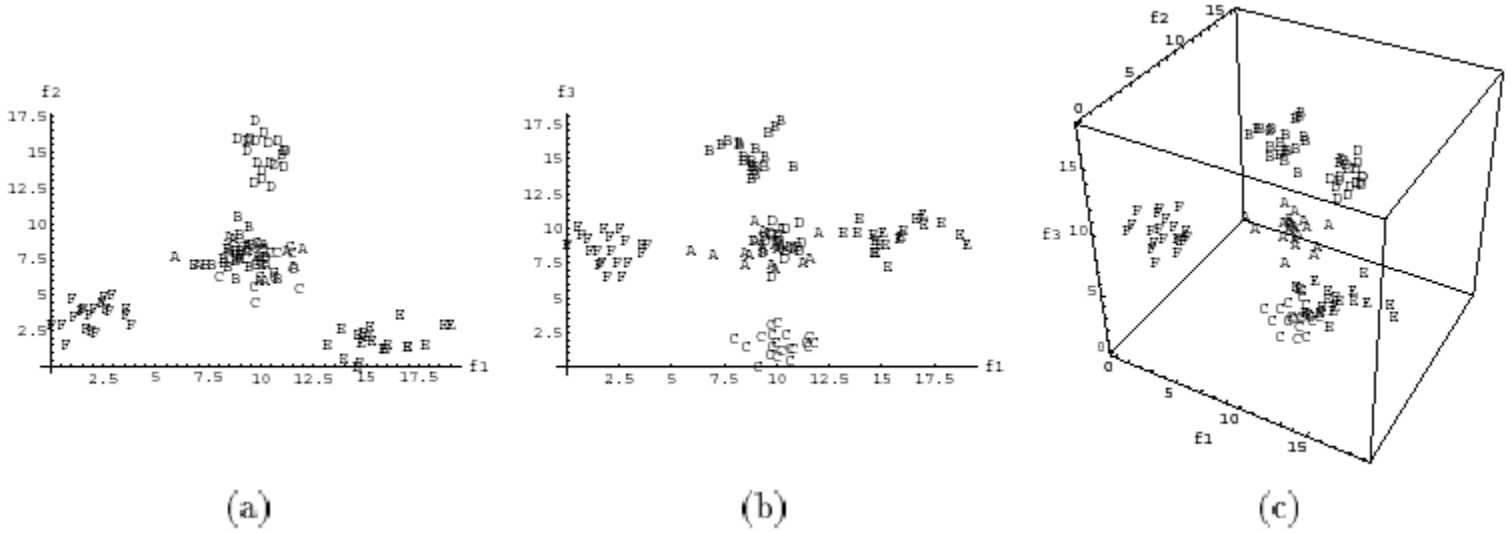


Comparison with MDS

- Response time vs. stress



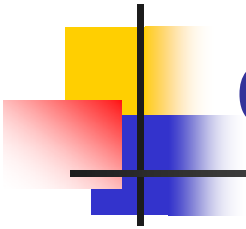
Clustering/visualization properties of FastMap



(a) (b) (c)

a) f2 vs f1 b) f3 vs f1 c)3-d plot (f1,f2,f3)

GAUSSIAN5D dataset



Conclusion

- A fast algorithm to map objects into points in k-d space
- Accelerate searching by highly optimized SAMs e.g. R-trees, R*-trees etc.
- Application of the algorithm to multimedia database, data-mining, clustering and document retrieval etc.