Retrieval techniques for high-dimensional datasets

- **The retrieval problem:**
  - Given a set of objects *S*, and a query object S,
  - find the objectss that are most similar to S.
- Applications:
  - financial, voice, marketing, medicine, video

1

Examples

- Find companies with similar stock prices over a time interval
- Find products with similar sell cycles
- Cluster users with similar credit card utilization
- Cluster products

2

Indexing when the triangle inequality holds

- Typical distance metric: L_p norm.
- We use L_2 as an example throughout:
  - $D(S,T) = \left(\Sigma_{i=1,..,n} (S[i] - T[i])^2\right)^{1/2}$

3

Indexing: The naïve way

- Each object is an n-dimensional tuple
- Use a high-dimensional index structure to index the tuples
- Such index structures include
  - R-trees,
  - kd-trees,
  - vp-trees,
  - grid-files...

4

High-dimensional index structures

- All require the triangle inequality to hold
- All partition either
  - the space or
  - the dataset into regions
- The objective is to:
  - search only those regions that could potentially contain good matches
  - avoid everything else

5

The naïve approach: Problems

- High-dimensionality:
  - decreases index structure performance (the curse of dimensionality)
  - slows down the distance computation
- Inefficiency

6

Dimensionality reduction

- The main idea: reduce the dimensionality of the space.
- Project the n-dimensional tuples that represent the time series in a k-dimensional space so that:
  - $k << n$
  - distances are preserved as well as possible

7

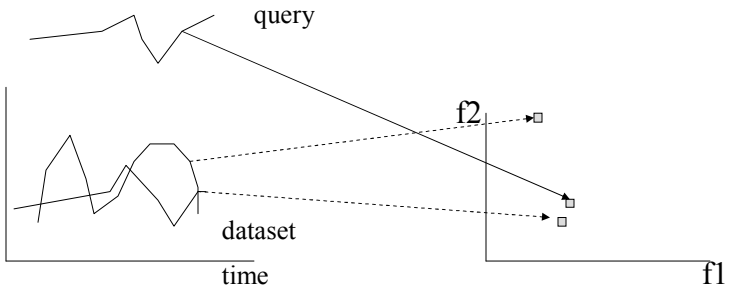
Dimensionality Reduction

- Use an indexing technique on the new space.
- GEMINI ([Faloutsos et al]):
  - Map the query S to the new space
  - Find nearest neighbors to S in the new space
  - Compute the actual distances and keep the closest

8

Dimensionality Reduction

- A time series is represented as a k-dim point
- The query is also transformed to the k-dim space



9

Dimensionality Reduction

- Let F be the dimensionality reduction technique:
  - Optimally we want:
  - D(F(S), F(T)) = D(S,T)
- Clearly not always possible.
- If D(F(S), F(T)) ≠ D(S,T)
  - false dismissal (when D(S,T) << D(F(S), F(T)))
  - false positives (when D(S,T) >> D(F(S), F(T)))

10

Dimensionality Reduction

- To guarantee no false dismissals we must be able to prove that:
  - $D(F(S),F(T)) < a\ D(S,T)$
  - for some constant a
- a small rate of false positives is desirable, but not essential

11

What we achieve

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

12

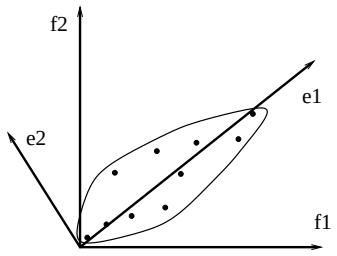
Dimensionality Techniques

- We will review a number of dimensionality techniques that can be applied in this context
  - SVD decomposition,
  - Discrete Fourier transform, and Discrete Cosine transform
  - Wavelets
  - Partitioning in the time domain
  - Random Projections
  - Multidimensional scaling
  - FastMap and its variants

13

SVD decomposition - the Karhunen-Loeve transform

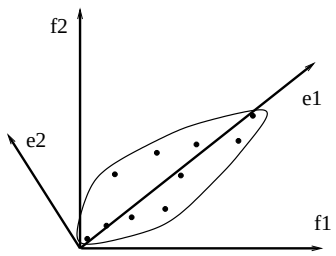
- Intuition: find the axis that shows the greatest variation, and project all points into this axis
- [Faloutsos, 1996]



14

SVD: The mathematical formulation

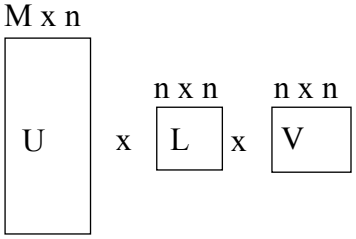
- Find the eigenvectors of the covariance matrix
- These define the new space
- The eigenvalues sort them in "goodness" order



15

SVD: The mathematical formulation, Cont'd

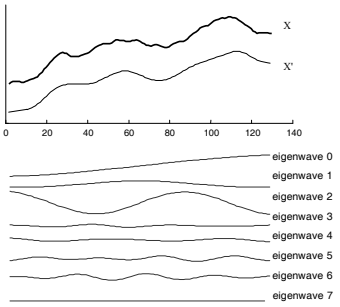
- Let A be the M x n matrix of M time series of length n
- The SVD decomposition of A is: = U x L x V^T,
  - U, V orthogonal
  - L diagonal
- L contains the eigenvalues of A^TA



16

SVD Cont'd

- To approximate the time series, we use only the k largest eigenvectors of C.
- A' = U x L_k
- A' is an M x k matrix



17

SVD Cont'd

- Advantages:
  - Optimal dimensionality reduction (for linear projections)
- Disadvantages:
  - Computationally hard, especially if the time series are very long.
  - Does not work for subsequence indexing

18

SVD Extensions

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

19

Discrete Fourier Transform

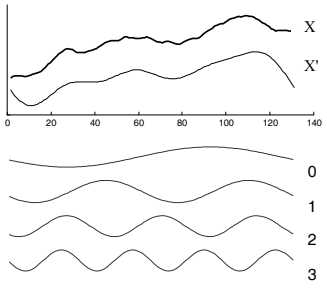
- Analyze the frequency spectrum of an one dimensional signal
- For $S = (S_0, \ldots, S_{n-1})$, the DFT is:
- $S_f = 1/\sqrt{n} \sum_{i=0,..,n-1} S_i \, e^{-j2\pi fi/n}$
  $f = 0,1,\ldots n-1$, $j^2 = -1$
- An efficient O(nlogn) algorithm makes DFT a practical method
- [Agrawal et al, 1993], [Rafiei and Mendelzon, 1998]

20

Discrete Fourier Transform

- To approximate the time series, keep the k largest Fourier coefficients only.
- Parseval's theorem:

  $\Sigma_{i=0,..,n-1} S_i^2 = \Sigma_{i=0,..,n-1} S_f^2$
- DFT is a linear transform so:
  - $\Sigma_{i=0,..,n-1}(S_i - T_i)^2 = \Sigma_{i=0,..,n-1}(S_f - T_f)^2$



21

Discrete Fourier Transform

- Keeping k DFT coefficients lower bounds the distance:
  - $\Sigma_{i=0,..,n-1}(S[i] - T[i])^2 > \Sigma_{i=0,..,k-1}(S_f - T_f)^2$
- Which coefficients to keep:
  - The first k (F-index, [Agrawal et al, 1993], [Rafiei and Mendelzon, 1998])
  - Find the optimal set (not dynamic) [R. Kanth et al, 1998]

22

Discrete Fourier Transform

- Advantages:
  - Efficient, concentrates the energy
- Disadvantages:
  - To project the n-dimensional time series into a k-dimensional space, the same k Fourier coefficients must be store for all series
  - This is not optimal for all series
  - To find the k optimal coefficients for M time series, compute the average energy for each coefficient

23

Wavelets

- Represent the time series as a sum of prototype functions like DFT
- Typical base used: Haar wavelets
- Difference from DFT: localization in time
- Can be extended to 2 dimensions
- [Chan and Fu, 1999]
- Has been very useful in graphics, approximation techniques

24

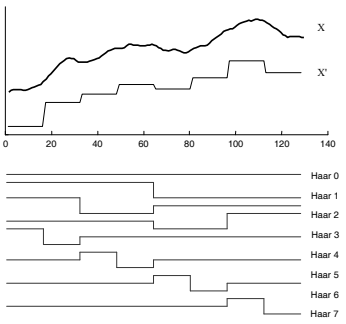
Wavelets

- An example (using the Haar wavelet basis)
  - S ≡ (2, 2, 7, 9) : original time series
  - S' ≡ (5, 6, 0, 2) : wavelet decomp.
  - S[0] = S'[0] - S'[1]/2 - S'[2]/2
  - S[1] = S'[0] - S'[1]/2 + S'[2]/2
  - S[2] = S'[0] + S'[1]/2 - S'[3]/2
  - S[3] = S'[0] + S'[1]/2 + S'[3]/2
- Efficient O(n) algorithm to find the coefficients

25

Using wavelets for approximation

- Keep only k coefficients, approximate the rest with 0
- Keeping the first k coefficients:
  - equivalent to low pass filtering
- Keeping the largest k coefficients:
  - More accurate representation, But not useful for indexing



26

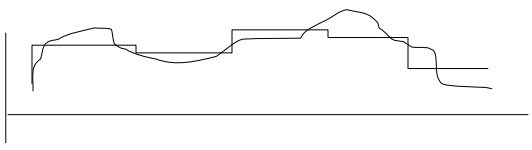
Wavelets

- Advantages:
  - The transformed time series remains in the same (temporal) domain
  - Efficient O(n) algorithm to compute the transformation
- Disadvantages:
  - Same with DFT

27

Line segment approximations

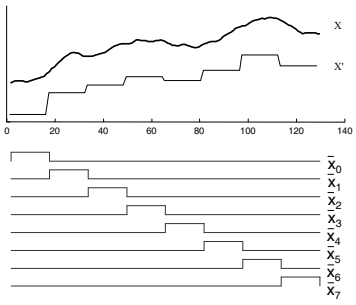
- Piece-wise Aggregate Approximation
  - Partition each time series into k subsequences (the same for all series)
  - Approximate each sequence by :
    - its mean and/or variance: [Keogh and Pazzani, 1999], [Yi and Faloutsos, 2000]
    - a line segment: [Keogh and Pazzani, 1998]



28

Temporal Partitioning

- Very Efficient technique (O(n) time algorithm)
- Can be extended to address the subsequence matching problem
- Equivalent to wavelets (when $k= 2^i$, and mean is used)



29

Random projection

- Based on the Johnson-Lindenstrauss lemma:
- For:
  - $0< e < 1/2$,
  - any (sufficiently large) set $\boldsymbol{S}$ of M points in R_n
  - $k = O(e^{-2}\ln M)$
- There exists a linear map $f:\boldsymbol{S} \rightarrow R_k$, such that
  - $(1-e)\, D(S,T) < D(f(S),f(T)) < (1+e)D(S,T)$ for S,T in $\boldsymbol{S}$
- Random projection is good with constant probability
- [Indyk, 2000]

30

Random Projection: Application

- Set k = $O(e^{-2}\ln M)$
- Select k random n-dimensional vectors
- Project the time series 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

31

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

32

Multidimensional 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…

33

Algorithms for MS

- Input: M time series, their pairwise distances, the desired dimensionality k.
- Optimization criterion:

  stress = $(\sum_{ij}(D(S_i,S_j) - D(S_{ki}, S_{kj}))^2 / \sum_{ij}D(S_i,S_j)^2)^{1/2}$

  – where $D(S_i,S_j)$ be the distance between time series S_i, S_j, and $D(S_{ki}, S_{kj})$ be the Euclidean distance of the k-dim representations
- Steepest descent algorithm:
  – start with an assignment (time series to k-dim point)
  – minimize stress by moving points

34

Multidimensional Scaling

- Advantages:
  - good dimensionality reduction results (though no guarantees for optimality
- Disadvantages:
  - How to map the query? O(M) obvious solution..
  - slow conversion algorithm

35

FastMap

[Faloutsos and Lin, 1995]

- Maps objects to k-dimensional points so that distances are preserved well
- It is an approximation of Multidimensional Scaling
- Works even when only distances are known
- Is efficient, and allows efficient query transformation

36

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

37

MetricMap
[Wang et al, 1999]

- Embeds objects into a k-dim pseudo-metric space
- Takes a random sample of points, and finds the eigenvectors of their covariance matrix
- Uses the larger eigenvalues to define the new k-dimensional space.
- Similar results to FastMap

38

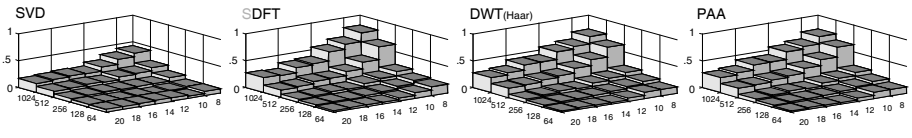
Dimensionality techniques: Summary

- SVD: optimal (for linear projections), slowest
- DFT: efficient, works well in certain domains
- Temporal Partitioning: most efficient, works well
- Random projection: very useful when applied with another technique
- FastMap: particularly useful when only distances are known

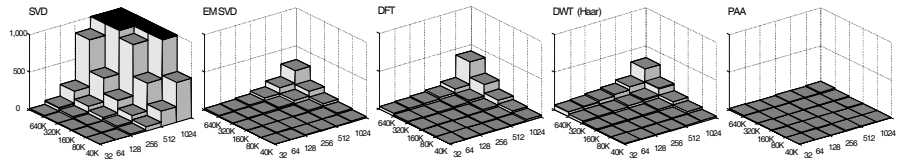
39

An experimental comparison of the techniques [Keogh et al, 2000]

- Accuracy:



- Speed of building the index:



40

Indexing Techniques

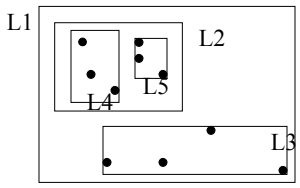
- We will look at:
  - R-trees and variants
  - kd-trees
  - vp-trees and variants
  - sequential scan
- R-trees and kd-trees partition the space,
  vp-trees and variants partition the dataset,
  there are also hybrid techniques

41

R-trees and variants

[Guttman, 1984], [Sellis et al, 1987], [Beckmann et al, 1990]

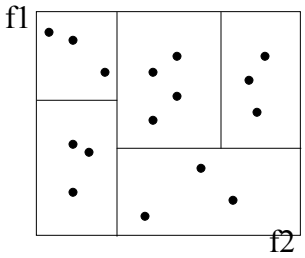
- k-dim extension of B-trees
- Balanced tree
- Intermediate nodes are rectangles that cover lower levels
- Rectangles may be overlapping or not depending on variant (R-trees, R+-trees, R*-trees)
- Can index rectangles as well as points



42

kd-trees

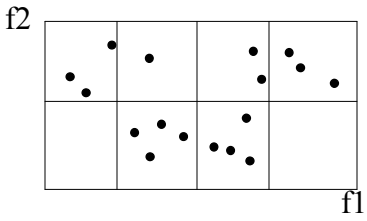
- Based on binary trees
- Different attribute is used for partitioning at different levels
- Efficient for indexing points
- External memory extensions: hB$^{\Pi}$-tree



43

Grid Files

- Use a regular grid to partition the space
- Points in each cell go to one disk page
- Can only handle points

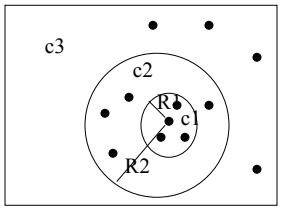


44

vp-trees and pyramid trees

[Ullmann], [Berchtold et al,1998], [Bozkaya et al1997],...

- Basic idea: partition the dataset, rather than the space
- vp-trees: At each level, partition the points based on the distance from a center
- Others: mvp-, TV-, S-, Pyramid-trees



The root level of a vp-tree with 3 children

45

Sequential Scan

- The simplest technique:
  - Scan the dataset once, computing the distances
  - Optimizations: give lower bounds on the distance quickly
  - Competitive when the dimensionality is large.

46

High-dimensional Indexing Methods: Summary

- For low dimensionality (<10), space partitioning techniques work best
- For high dimensionality, sequential scan will probably be competitive with any technique
- In between, dataset partitioning techniques work best

47

Open problems

- Indexing non-metric distance functions
- Similarity models and indexing techniques for higher-dimensional time series
- Efficient trend detection/subsequence matching algorithms

48