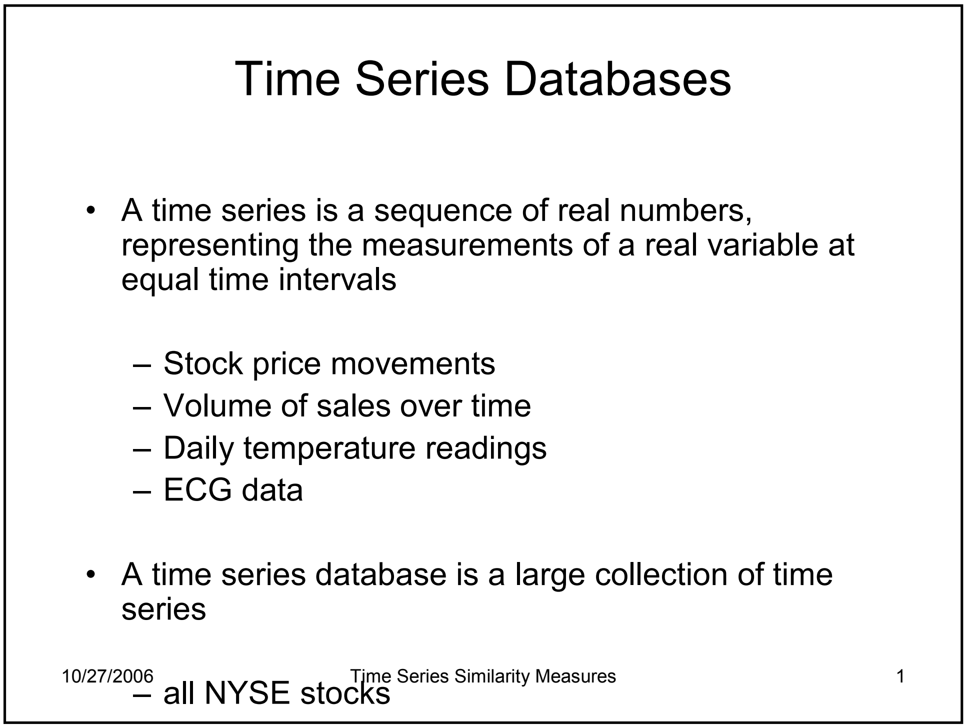
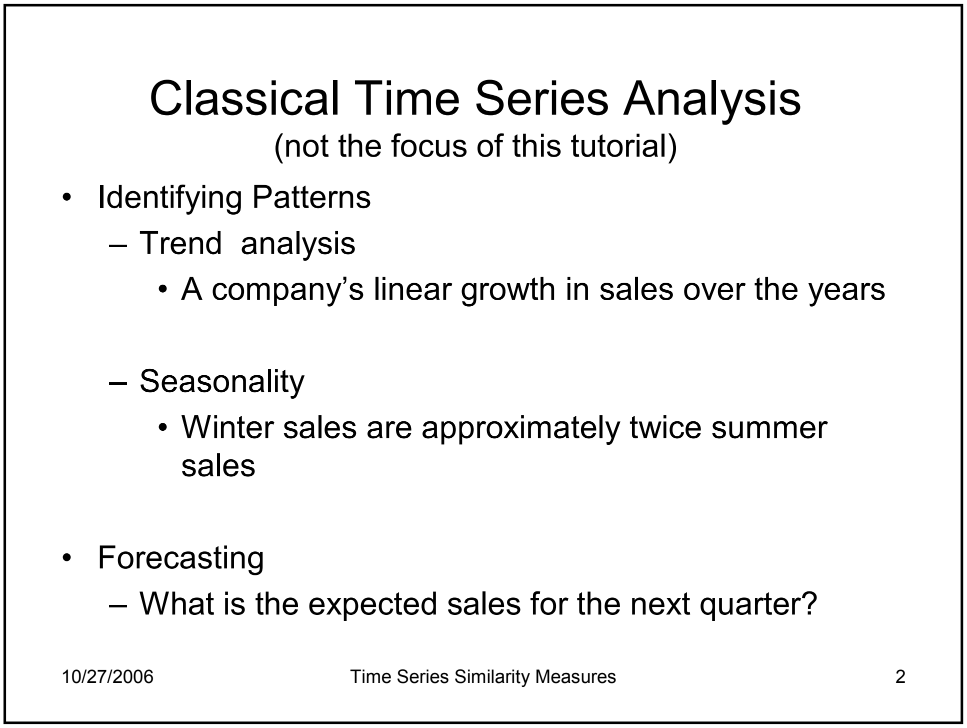


Time Series Databases

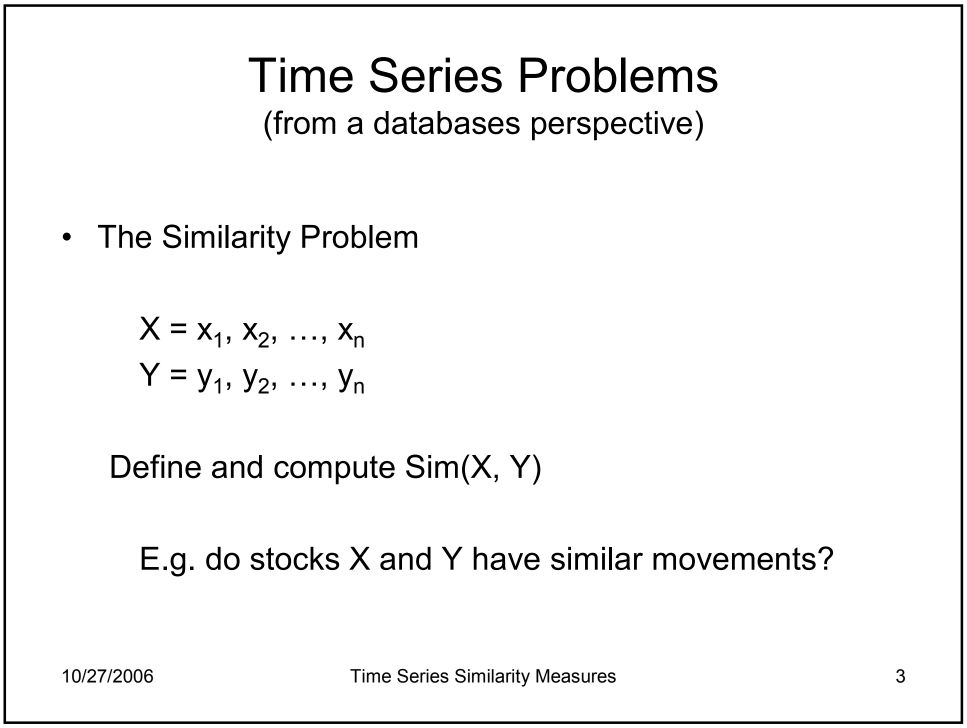
- A time series is a sequence of real numbers, representing the measurements of a real variable at equal time intervals
  - – Stock price movements
  - – Volume of sales over time
  - – Daily temperature readings
  - – ECG data
- A time series database is a large collection of time series
  - – all NYSE stocks



Classical Time Series Analysis

(not the focus of this tutorial)

- Identifying Patterns
  - – Trend analysis
    - A company's linear growth in sales over the years
  - – Seasonality
    - Winter sales are approximately twice summer sales
- Forecasting
  - – What is the expected sales for the next quarter?



Time Series Problems

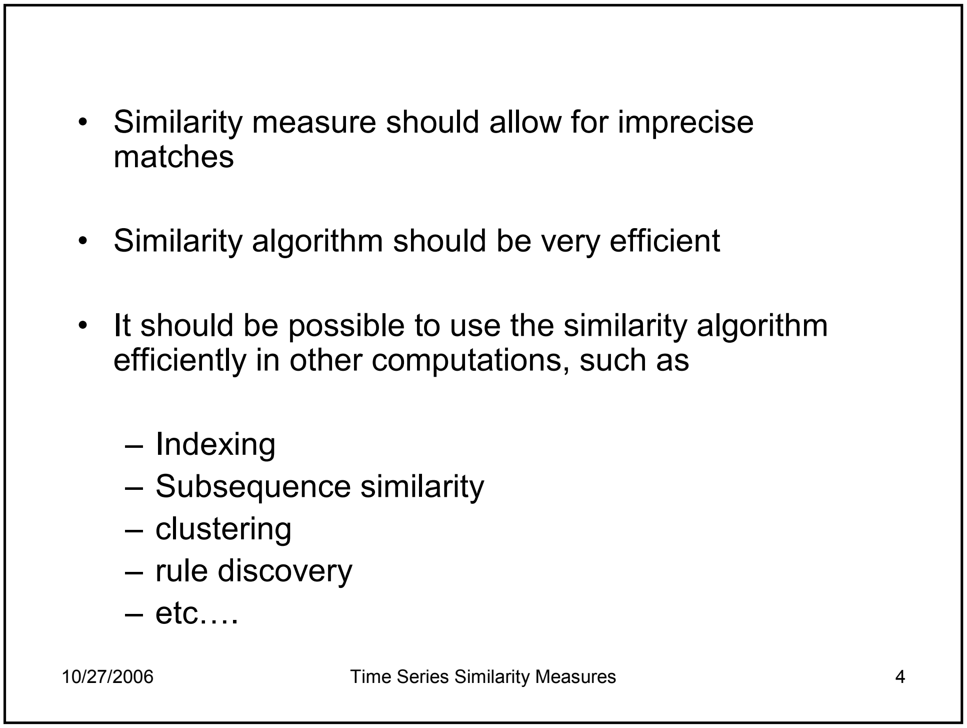
(from a databases perspective)

- The Similarity Problem

$X = x_1, x_2, \ldots, x_n$
$Y = y_1, y_2, \ldots, y_n$

Define and compute Sim(X, Y)

E.g. do stocks X and Y have similar movements?



- Similarity measure should allow for imprecise matches
- Similarity algorithm should be very efficient
- It should be possible to use the similarity algorithm efficiently in other computations, such as
  - Indexing
  - Subsequence similarity
  - clustering
  - rule discovery
  - etc.…

- Indexing problem
  - Find all lakes whose water level fluctuations are similar to X
- Subsequence Similarity Problem
  - Find out other days in which stock X had similar movements as today
- Clustering problem
  - Group regions that have similar sales patterns
- Rule Discovery problem
  - Find rules such as “if stock X goes up and Y remains the same, then Z will shortly go down”

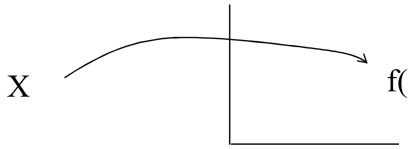
Examples

- 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
- Use patterns to classify a given time series
- Find patterns that are frequently repeated
- Find similar subsequences in DNA sequences
- Find scenes in video streams

- Basic approach to the Indexing problem

Extract a few key “features” for each time series

Map each time sequence X to a point f(X) in the (relatively low dimensional) “feature space”, such that the (dis) similarity between X and Y is approximately equal to the Euclidean distance between the two points f(X) and f(Y)



Use any well-known spatial access method (SAM) for indexing the feature space

- Scalability an important issue
  - If similarity measures, time series models, etc. become more sophisticated, then the other problems (indexing, clustering, etc.) become prohibitive to solve
- Research challenge
  - Design solutions that attempt to strike a balance between accuracy and efficiency

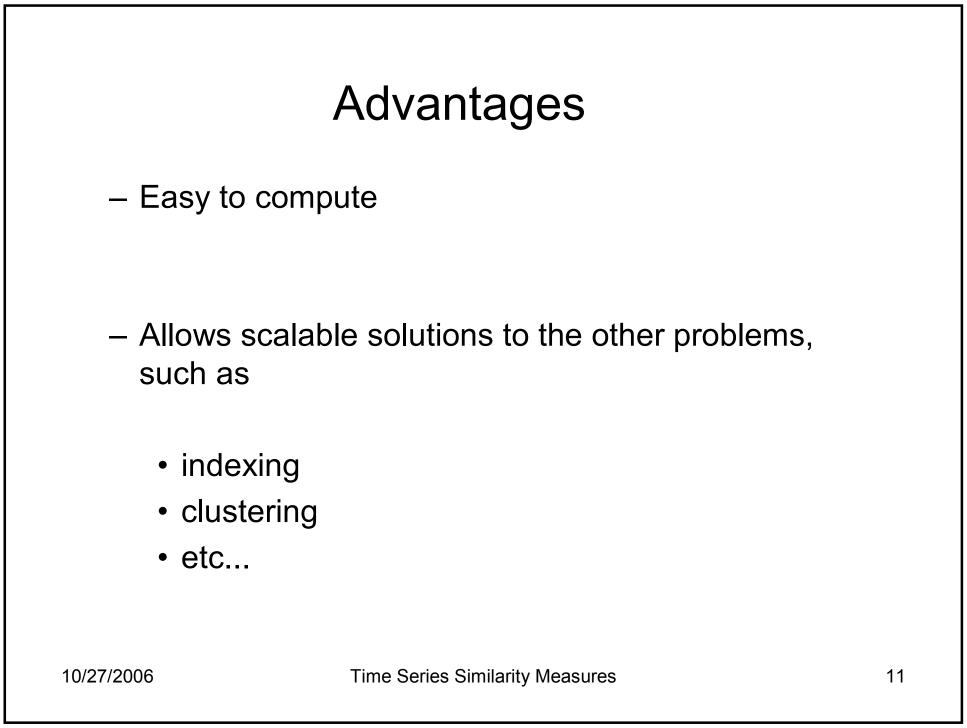
Outline of Tutorial

- Part I
  - Discussion of various similarity measures
- Part II
  - Discussion of various solutions to the other problems, such as indexing, subsequence similarity, etc
  - Query language support for time series
  - Miscellaneous issues ...

Euclidean Similarity Measure

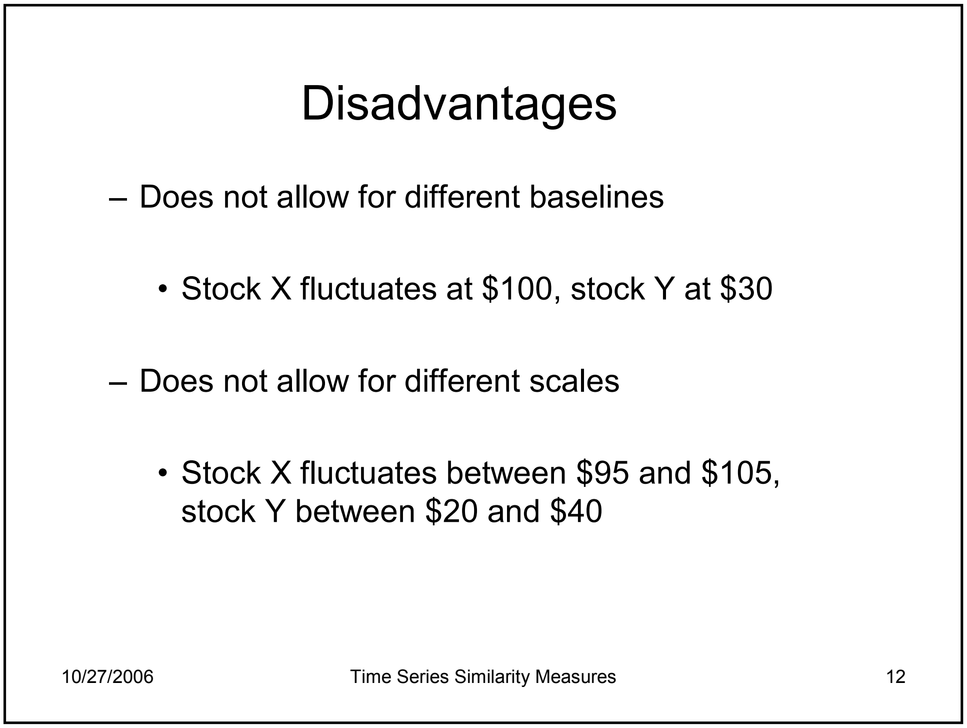
- View each sequence as a point in n-dimensional Euclidean space (n = length of sequence)
- Define (dis)similarity between sequences X and Y as

$$Lp\ (X, Y)$$



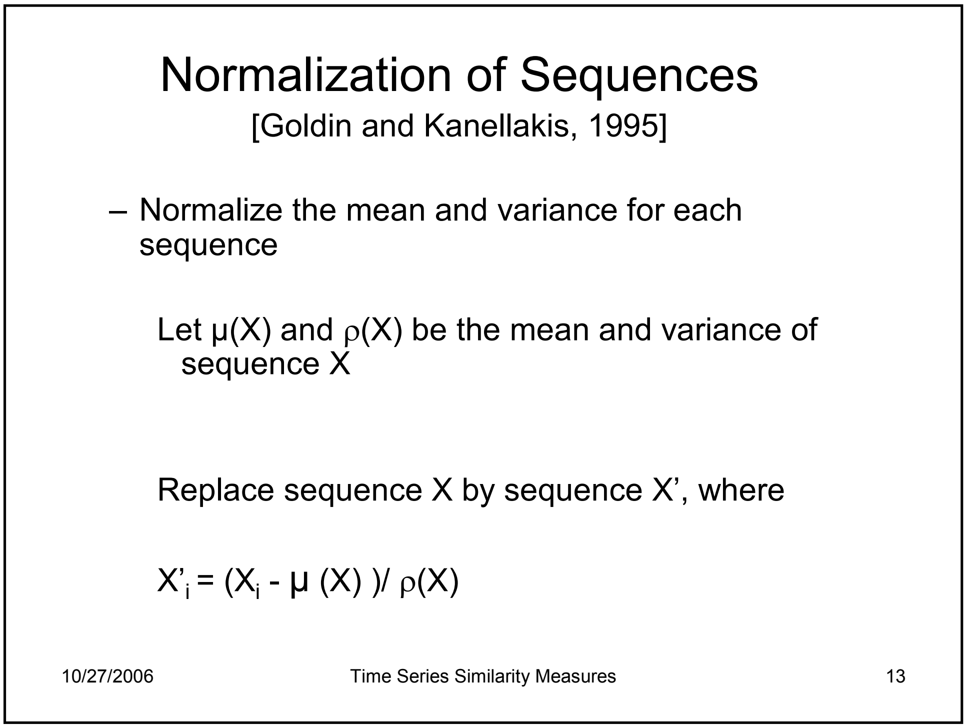
Advantages

- Easy to compute
- Allows scalable solutions to the other problems, such as
  - indexing
  - clustering
  - etc...



Disadvantages

- Does not allow for different baselines
  - Stock X fluctuates at $100, stock Y at $30
- Does not allow for different scales
  - Stock X fluctuates between $95 and $105, stock Y between $20 and $40



Normalization of Sequences

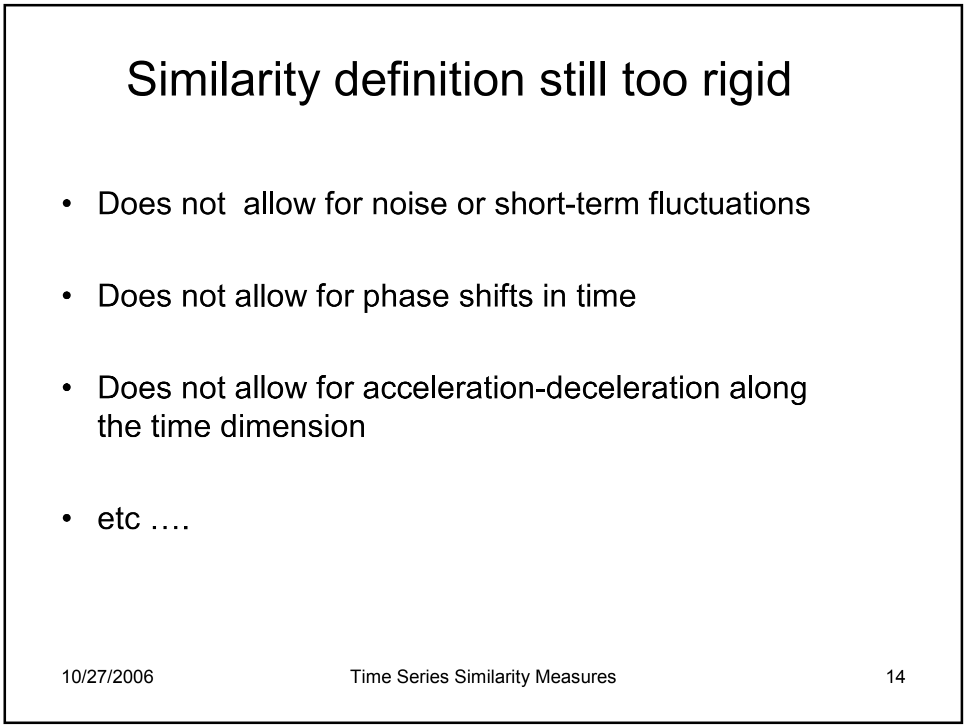
[Goldin and Kanellakis, 1995]

– Normalize the mean and variance for each sequence

Let $\mu(X)$ and $\rho(X)$ be the mean and variance of sequence X

Replace sequence X by sequence X’, where

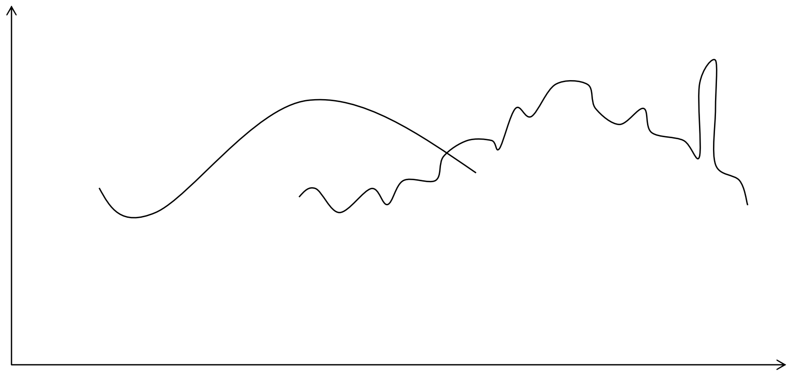
$X'_i = (X_i - \mu\ (X)\ )/\ \rho(X)$



Similarity definition still too rigid

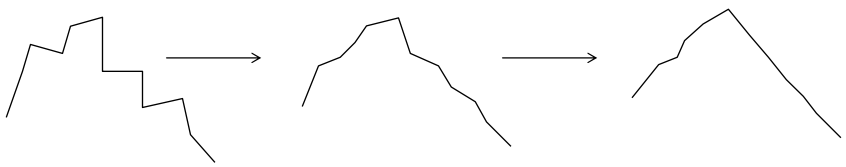
- Does not allow for noise or short-term fluctuations
- Does not allow for phase shifts in time
- Does not allow for acceleration-deceleration along the time dimension
- etc ….

Example



A general similarity framework involving a transformation rules language

[Jagadish, Mendelzon, Milo]

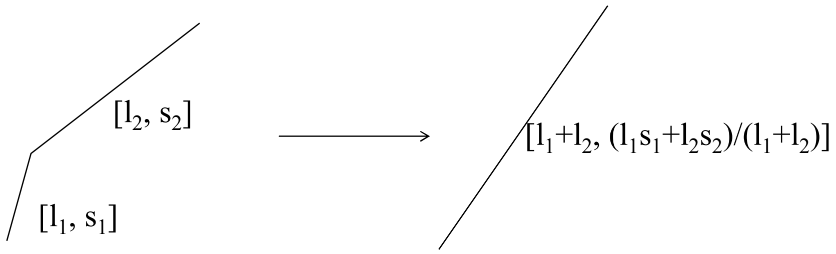


Each rule has an associated cost

Examples of Transformation Rules

- Collapse adjacent segments into one segment

new slope = weighted average of previous slopes
new length = sum of previous lengths



Combinations of Moving Averages, Scales, and Shifts

[Rafiei and Mendelzon, 1998]

– Moving averages are a well-known technique for smoothening time sequences

- Example of a 3-day moving average
  $x'_i = (x_{i-1} + x_i + x_{i+1})/3$

Disadvantages of Transformation Rules

- Subsequent computations (such as the indexing problem) become more complicated
  - Feature extraction becomes difficult, especially if the rules to apply become dependent on the particular X and Y in question
  - Euclidean distances in the feature space may not be good approximations of the sequence distances in the original space

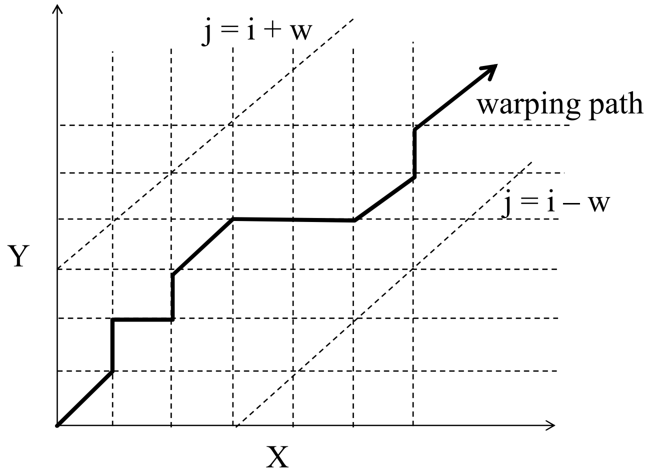
Dynamic Time Warping

[Berndt, Clifford, 1994]

- Extensively used in speech recognition
- Allows acceleration-deceleration of signals along the time dimension
- Basic idea
  - Consider X = $x_1, x_2, \ldots, x_n$, and Y = $y_1, y_2, \ldots, y_n$
  - We are allowed to extend each sequence by repeating elements
  - Euclidean distance now calculated between the extended sequences X’ and Y’

Dynamic Time Warping

[Berndt, Clifford, 1994]



Restrictions on Warping Paths

- Monotonicity
  - Path should not go down or to the left
- Continuity
  - No elements may be skipped in a sequence
- Warping Window

  | i – j | <= w
- Others ….

Formulation

- Let D(i, j) refer to the dynamic time warping distance between the subsequences

  $x_1, x_2, \ldots, x_i$
  $y_1, y_2, \ldots, y_j$

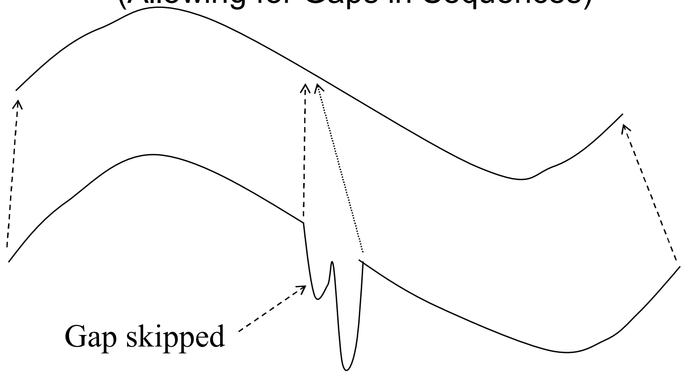
  $D(i, j) = | x_i – y_j | + \min \{ D(i – 1, j), D(i – 1, j – 1), D(i, j – 1) \}$

Solution by Dynamic Programming

- Basic implementation = $O(n^2)$ where n is the length of the sequences
  - will have to solve the problem for each (i, j) pair
- If warping window is specified, then O(nw)
  - Only solve for the (i, j) pairs where | i – j | <= w

Longest Common Subsequence Measures

(Allowing for Gaps in Sequences)



Basic LCS Idea

X	=	3, **2**, **5**, **7**, 4, 8, **10**, 7
Y	=	**2**, **5**, 4, **7**, 3, **10**, 8, 6
LCS	=	**2**, **5**, **7**, **10**

Sim(X,Y) = |LCS|

Shortcomings

Different scaling factors and baselines (thus need to scale, or transform one sequence to the other)

Should allow tolerance when comparing elements (even after transformation)

- Longest Common Subsequences
  - Often used in other domains
    - Speech Recognition
    - Text Pattern Matching
  - Different flavors of the LCS concept
    - Edit Distance

LCS-like measures for time series

- Subsequence comparison without scaling [Yazdani & Ozsoyoglu, 1996]
- Subsequence comparison with local scaling and baselines [Agrawal et. al., 1995]
- Subsequence comparision with global scaling and baselines [Das et. al., 1997]
- Global scaling and shifting [Chu and Wong, 1999]

LCS without Scaling

[Yazdani & Ozsoyoglu, 1996]

Let Sim(i, j) refer to the similarity between the sequences

$x_1, x_2, \ldots, x_i$ and $y_1, y_2, \ldots .y_j$

Let d be an allowed tolerance, called the "threshold distance"

If $| x_i - y_j | < d$ then

$\mathrm{Sim}(i, j) = 1 + D(i – 1, j - 1)$

else $\mathrm{Sim}(i, j) = \max\{D(i – 1, j), D(i, j – 1)\}$

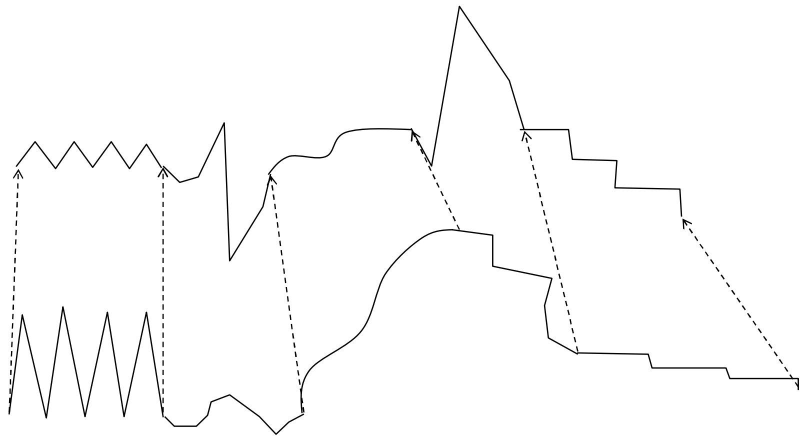
LCS-like Similarity with Local Scaling

[Agrawal et al, 1995]

- Basic Ideas
  - Two sequences are similar if they have enough non-overlapping time-ordered pairs of subsequences that are similar
  - A pair of subsequences are similar if one can be scaled and translated appropriately to approximately resemble the other

Three pairs of subsequences

Scale & translation different for each pair



The Algorithm

- Find all pairs of atomic subsequences in X and Y that are similar
  - atomic implies of a certain minimum size (say, a parameter w)
- Stitch similar windows to form pairs of larger similar subsequences
- Find a non-overlapping ordering of subsequence matches having the longest match length

LCS-like Similarity with Global Scaling

[Das, Gunopulos and Mannila, 1997]

- Basic idea: Two sequences X and Y are similar if they have long common subsequence X' and Y' such that

  Y' is approximately = aX' + b

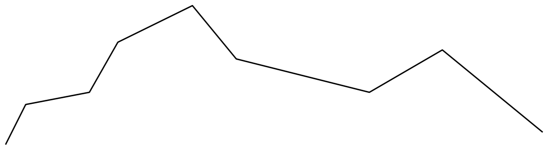
- The scale+translation linear function is derived from the subsequences, and not from the original sequences
  - Thus outliers cannot taint the scale+translation function
- Algorithm
  - Linear-time randomized approximation algorithm

- Main task for computing Sim
  - Locate a finite set of all *fundamentally different* linear functions
  - Run a dynamic-programming algorithm using each linear function

- Of the total possible linear functions, a constant fraction of them are *almost as good* as the optimal function

- The algorithm just picks a few (constant) number of functions at random and tries them out

Piecewise Linear Representation of Time Series



Time series approximated by K linear segments

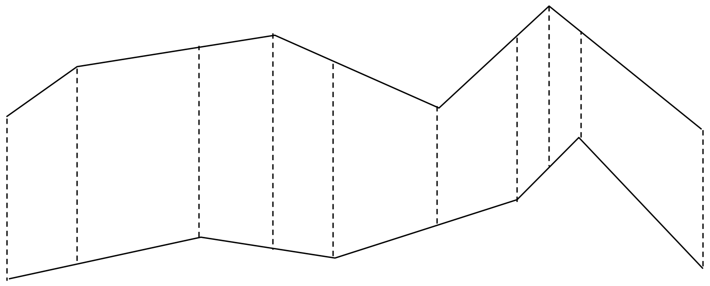
- Such approximation schemes
  - achieve data compression
  - allow scaling along the time axis

- How to select K?
  - Too small => many features lost
  - Too large => redundant information retained

- Given K, how to select the best-fitting segments?
  - Minimize some error function

- These problems pioneered in [Pavlidis & Horowitz 1974], further studied by [Keogh, 1997]

Defining Similarity



Distance = (weighted) sum of the difference of projected segments [Keogh & Pazzani, 1998]

Probabilistic Approaches to Similarity

[Keogh & Smyth, 1997]

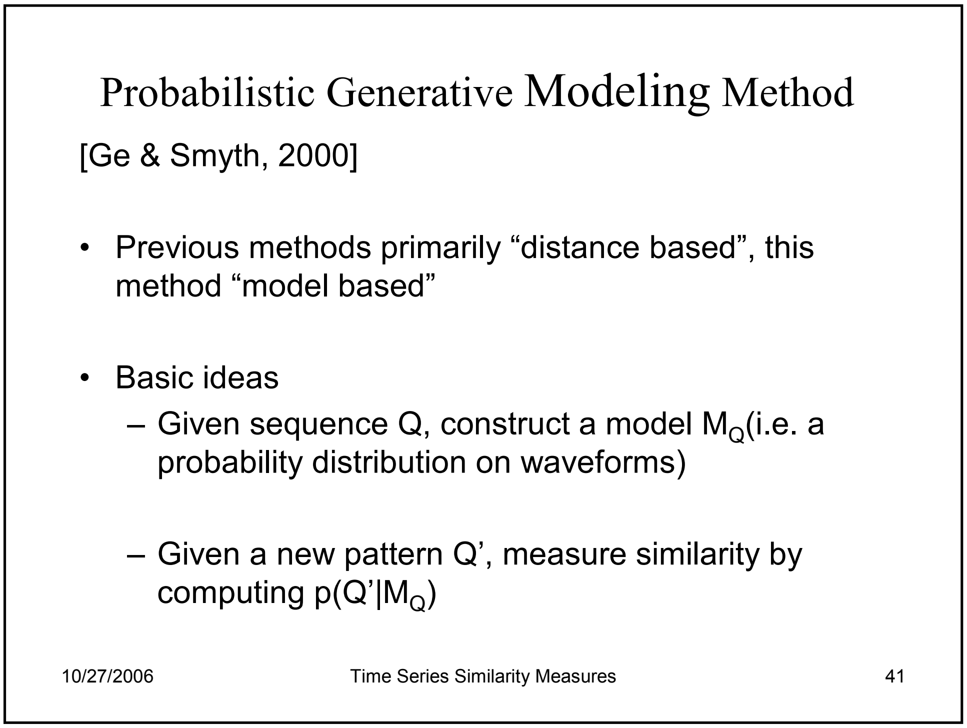
- Probabilistic distance model between time series Q and R
  - Ideal template Q which can be “deformed” (according to a prior distribution) to generate the the observed data R
  - If D is the observed deformation between Q and R, we need to define the generative model

    p(D | Q)

- Piecewise linear representation of time series R
- Query Q represented as
  - a sequence of local features (e.g. peaks, troughs, plateaus) which can be deformed according to prior distributions
  - global shape information represented as another prior on the relative location of the local features

Properties of the Probabilistic Measure

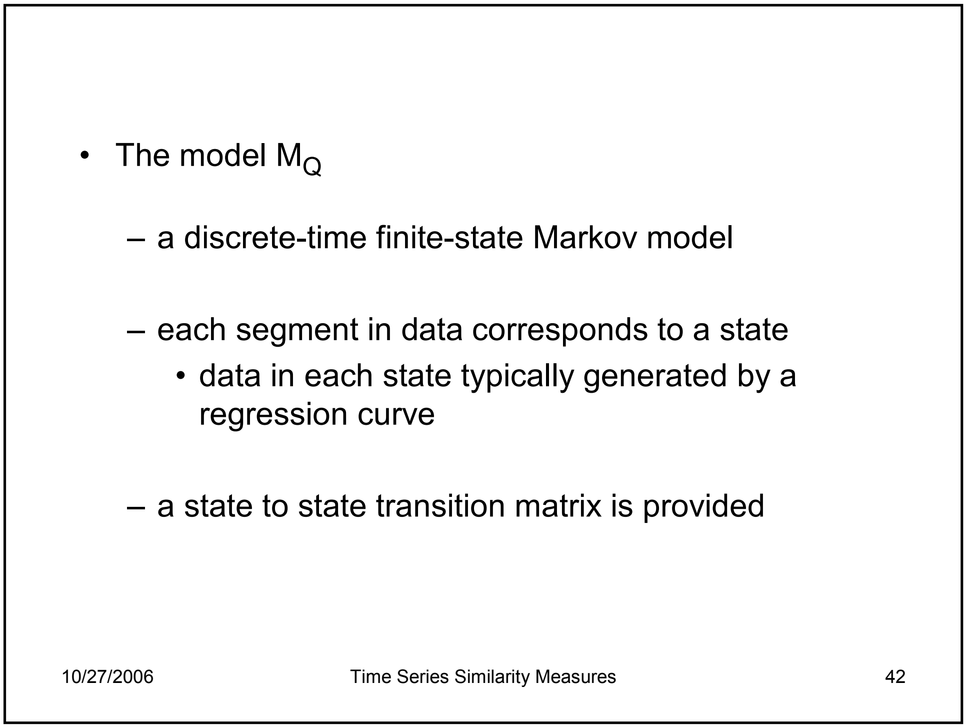
- Handles scaling and offset translations
- Incorporation of prior knowledge into similarity measure
- Handles noise and uncertainty



Probabilistic Generative Modeling Method

[Ge & Smyth, 2000]

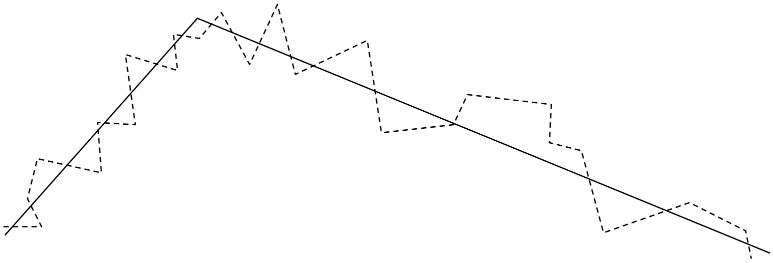
- Previous methods primarily “distance based”, this method “model based”
- Basic ideas
  - Given sequence Q, construct a model M_Q(i.e. a probability distribution on waveforms)
  - Given a new pattern Q’, measure similarity by computing $p(Q'|M_Q)$



- The model M_Q
  - a discrete-time finite-state Markov model
  - each segment in data corresponds to a state
    - data in each state typically generated by a regression curve
  - a state to state transition matrix is provided

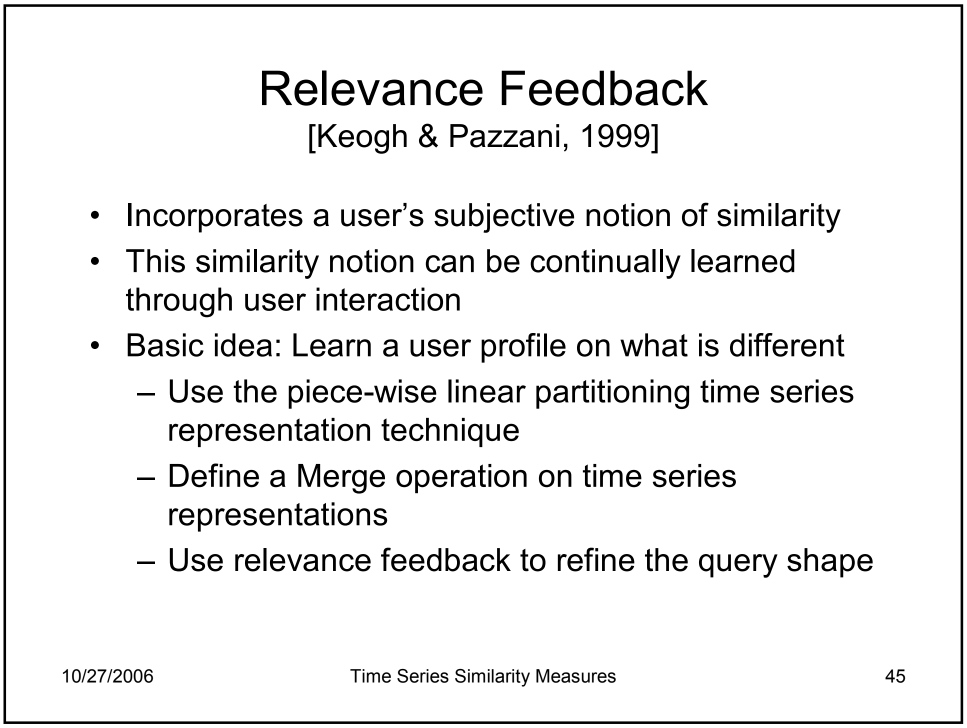
- On entering state i, a duration t is drawn from a state-duration distribution p(t)
  - the process remains in state i for time t
  - after this, the process transits to another state according to the state transition matrix

Example: output of Markov Model



Solid lines: the two states of the model

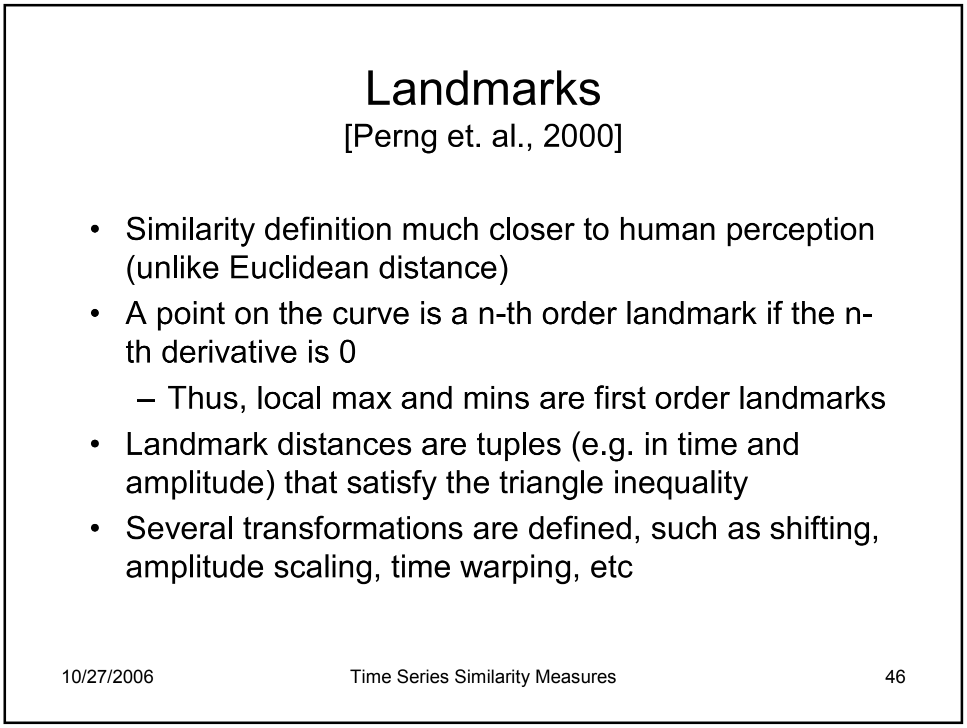
Dashed lines: the actual noisy observations



Relevance Feedback

[Keogh & Pazzani, 1999]

- Incorporates a user’s subjective notion of similarity
- This similarity notion can be continually learned through user interaction
- Basic idea: Learn a user profile on what is different
  - Use the piece-wise linear partitioning time series representation technique
  - Define a Merge operation on time series representations
  - Use relevance feedback to refine the query shape



Landmarks

[Perng et. al., 2000]

- Similarity definition much closer to human perception (unlike Euclidean distance)
- A point on the curve is a n-th order landmark if the n-th derivative is 0
  - Thus, local max and mins are first order landmarks
- Landmark distances are tuples (e.g. in time and amplitude) that satisfy the triangle inequality
- Several transformations are defined, such as shifting, amplitude scaling, time warping, etc

Retrieval techniques for time-series

- **The Time series retrieval problem:**
  - Given a set of time series *S*, and a query time series S,
  - find the series that are more similar to S.

- Applications:
  - Time series clustering for:
    financial, voice, marketing, medicine, video
  - Identifying trends
  - Nearest neighbor classification

The setting

- Sequence matching or subsequence matching
- Distance metric
- Nearest neighbor queries,
  range queries,
  all-pairs nearest neighbor queries

Retrieval algorithms

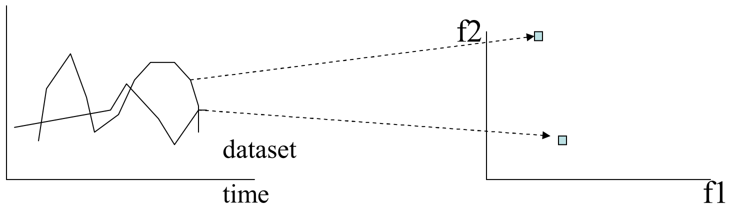
- We mainly consider the following setting:
  - the similarity function obeys the triangle inequality: D(A,B) < D(A,C) + D(C,B).
  - the query is a full length time series
  - we solve the nearest neighbor query
- We briefly examine the other problems: no distance metric, subsequence matching, all-pairs nearest neighbors

Indexing sequences when the triangle inequality holds

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

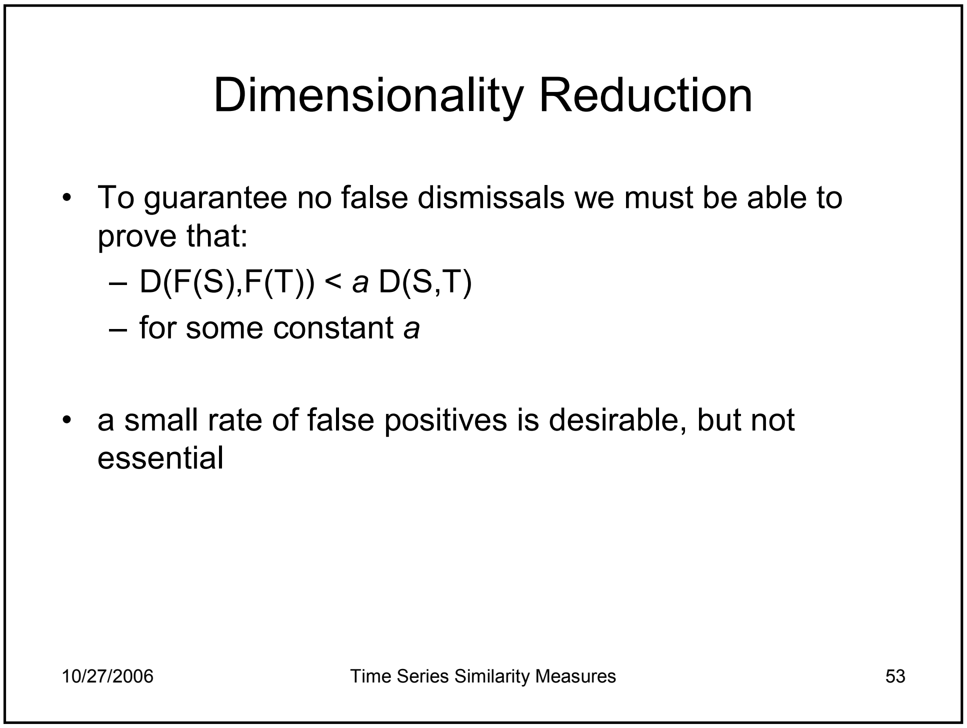
Dimensionality reduction

- 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



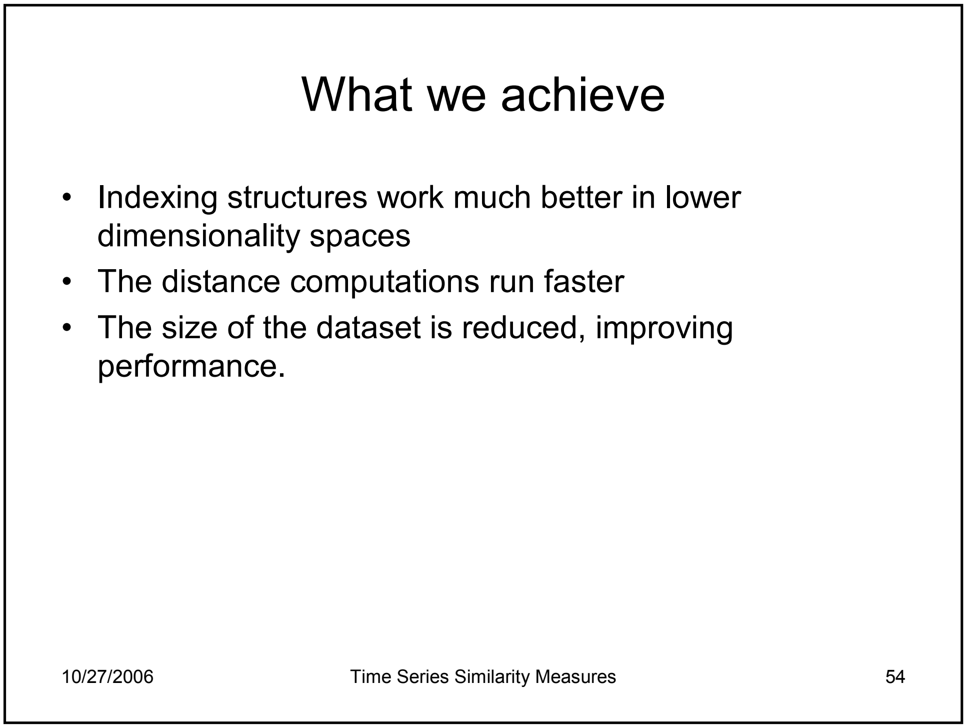
Dimensionality Reduction

- 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



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



What we achieve

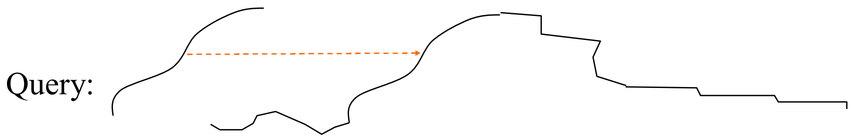
- Indexing structures work much better in lower dimensionality spaces
- The distance computations run faster
- The size of the dataset is reduced, improving performance.

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

The subsequence matching problem

- There is less work on this area
- The problem is more general and difficult
- [Faloutsos et al, 1994] [Park et al, 2000] [Kahveci, Singh, 2001] [Moon, Whang, Loh, 2001]
- Most of the previous dimensionality reduction techniques cannot be extended to handle the subsequence matching problem



The subsequence matching problem

- If the length of the subsequence is known, two general techniques can be applied:
  - Index all possible subsequences of given length k
    - n-w+1 subsequences of length w for each time series of length n
  - Partition each time series into fewer subsequences, and use an approximate matching retrieval mechanism

Similar sequence retrieval when triangle inequality doesn’t hold

- In this case indexing techniques do not work (except for sequential scan)
- Most techniques try to speed up the sequential scan by bounding the distance from below.

Distance bounding techniques

- Use a dimensionality reduction technique that needs only distances (FastMap, MetricMap, MS)
- Use a pessimistic estimate to bound the actual distance (and possibly accept a number of false dismissals)

  [Kim, Park, and Chu, 2001]
- Index the time series dataset using the reduced dimensionality space

Example: Time warping and FastMap

[Yi et al, 1998]

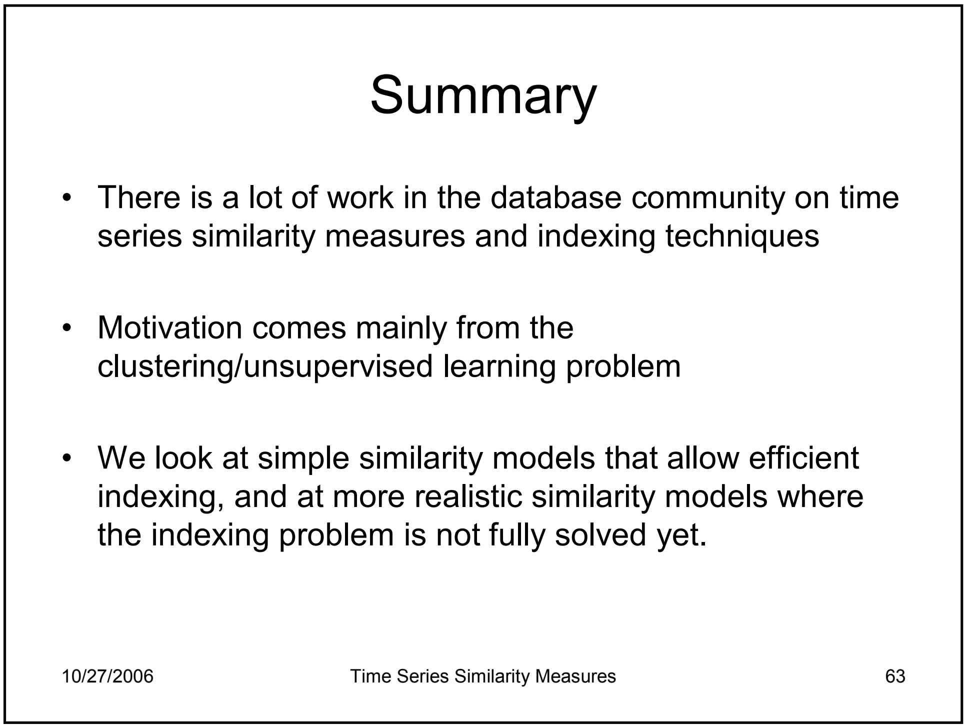
- Given M time series
  - Find the M(M-1)/2 distances using the time warping distance measure (does not satisfy the triangle inequality)
  - Use FastMap to project the time series to a k-dim space
- Given a query time series S,
  - Find the closest time series in the FastMap space
  - Retrieve them, and find the actual closest among them
- A heuristic technique: There is no guarantee that false dismissals are avoided

Indexing sequences of images

- When indexing sequences of images, similar ideas apply:
  - If the similarity/distance criterion is a metric, Use a dimensionality reduction technique
- [Yadzani and Ozsoyoglu]:
  - Map each image to a set of N features
  - Use a Longest Common Subsequence distance metric to find the distance between feature sequences
  - sim(ImageA, ImageB) = $\sum_{i=1..N}$sim(FA_i - FB_i)
- [Lee et al, 2000]:
  - Time warping distance measure
  - Use of Minimum Bounding Rectangles to lower bound the distance

Open problems

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



Summary

- There is a lot of work in the database community on time series similarity measures and indexing techniques
- Motivation comes mainly from the clustering/unsupervised learning problem
- We look at simple similarity models that allow efficient indexing, and at more realistic similarity models where the indexing problem is not fully solved yet.