



Why are Dendrograms Useful?

If someone tells us they have a new similarity measure for DNA, and it produces an intuitive dendrogram...

...but if their new similarity measure gives us a very un intuitive dendrogram, we should view it with suspicion...

Piecewise Linear Approximation I

Basic Idea: Represent the time series as a sequence of straight lines.

Lines could be **connected**, in which case we are allowed $N/2$ lines

If lines are **disconnected**, we are allowed only $N/3$ lines

Personal experience on dozens of datasets suggest **disconnected** is better. Also only **disconnected** allows a lower bounding Euclidean approximation

Each line segment has

- length
- left_height
- right_height

(right_height can be inferred by looking at the next segment)

Each line segment has

- length
- left_height
- right_height

Indexování a dobývání znalostí z časových řad (*time series*)

Eamonn Keogh

eamonn@cs.ucr.edu

Defining Distance Measures

Definition: Let O_1 and O_2 be two objects from the universe of possible objects. The distance (dissimilarity) is denoted by $D(O_1, O_2)$

What properties are desirable in a distance measure?

- $D(A,B) = D(B,A)$ Symmetry
- $D(A,A) = 0$ Constancy
- $D(A,B) = 0$ Iff $A=B$ Positivity
- $D(A,B) \leq D(A,C) + D(B,C)$ Triangular Inequality



Lowland Gorilla
Gorilla gorilla gorilla

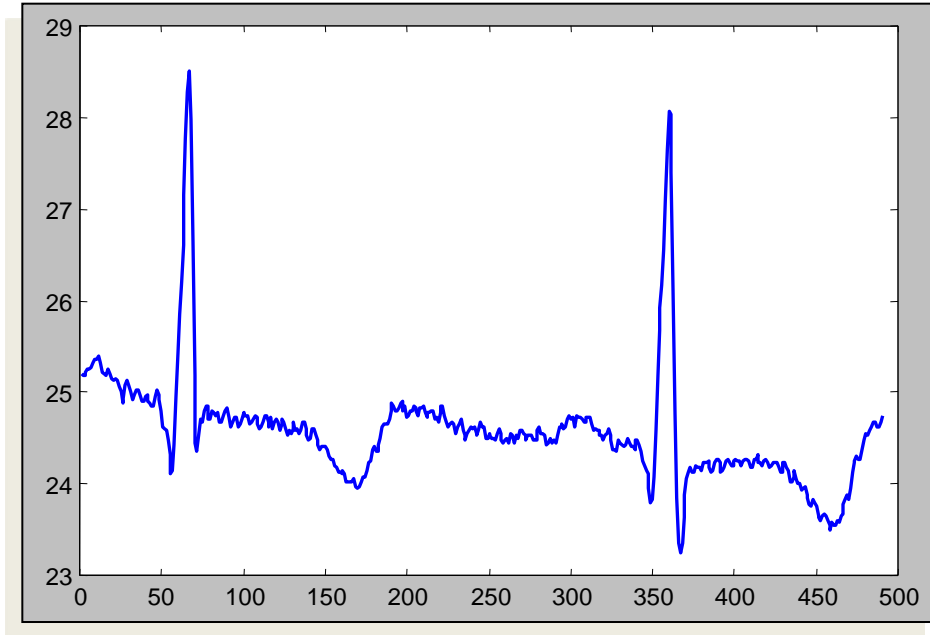
Mountain Gorilla
Gorilla gorilla beringei

DTW is needed for most natural objects...

25.1750
25.2250
25.2500
25.2500
25.2750
25.3250
25.3500
25.3500
25.4000
25.4000
25.3250
25.2250
25.2000
25.1750
..
..
24.6250
24.6750
24.6750
24.6250
24.6250
24.6250
24.6750
24.7500

What are Time Series?

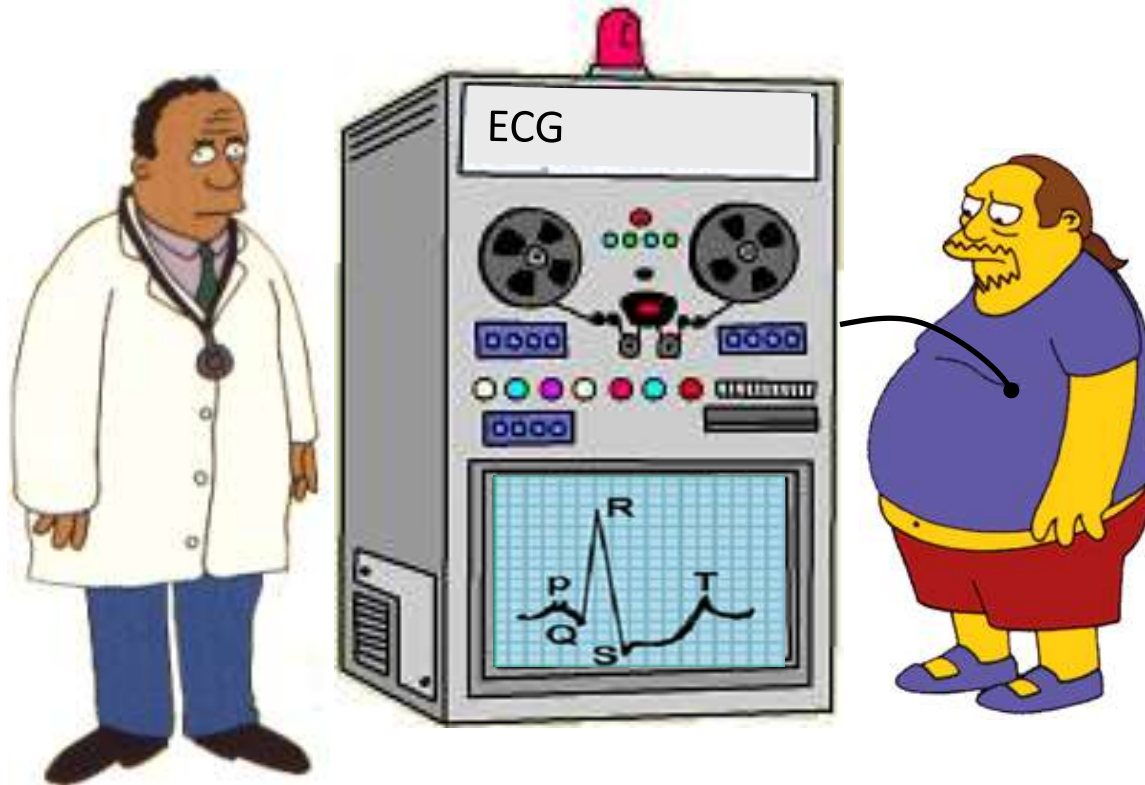
A time series is a collection of observations made sequentially in time.



Virtually all similarity measurements, indexing and dimensionality reduction techniques discussed in this tutorial can be used with other data types



Motivating example ...



You go to the doctor because of chest pains. Your ECG looks strange...

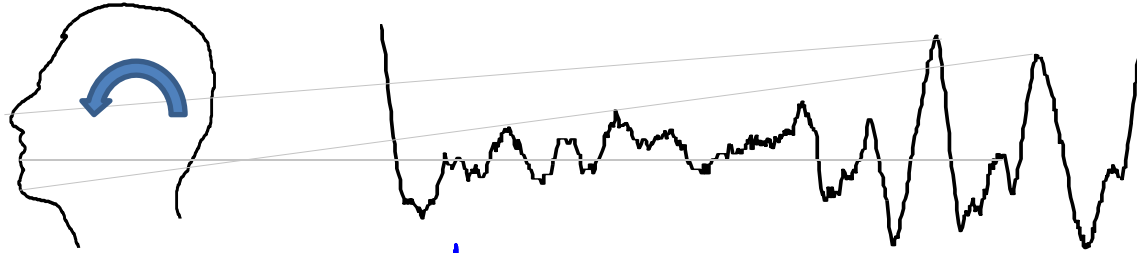
Your doctor wants to search a database to find **similar** ECGs, in the hope that they will offer clues about your condition...

Two questions:

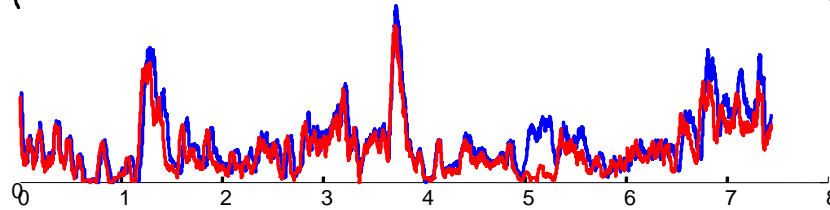
- How do we define similar? – distance measures
- How do we find it quickly?

What else can be viewed and processed in a similar way?

a) Images,
videos, ...



b) texts

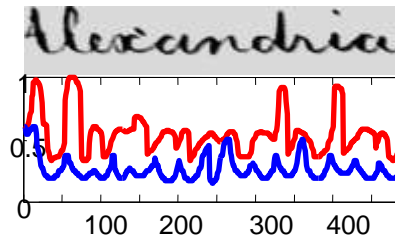


“God” - English Bible
“Dios” - Spanish Bible



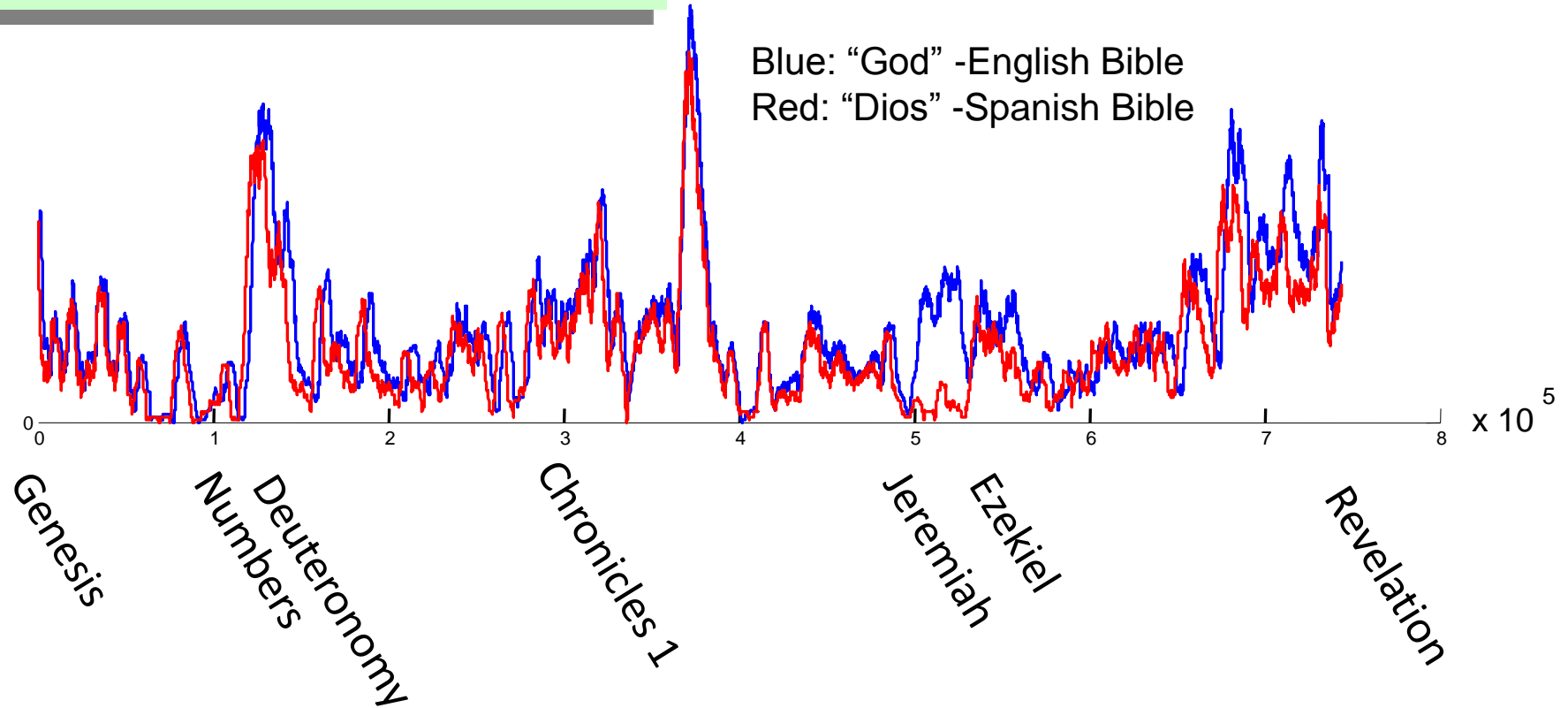
“El Senor” -
Spanish Bible

c) writing



Text data, may best be thought of as time series...

The local frequency of words in the Bible



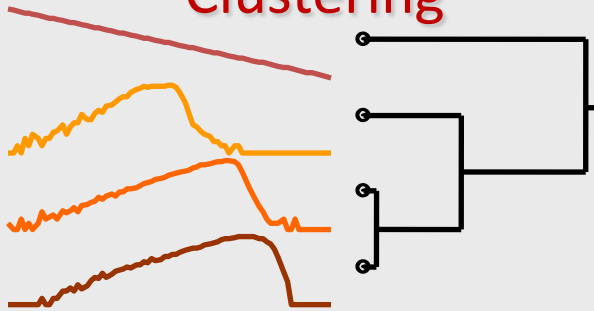
Gray: "El Senor" -Spanish Bible



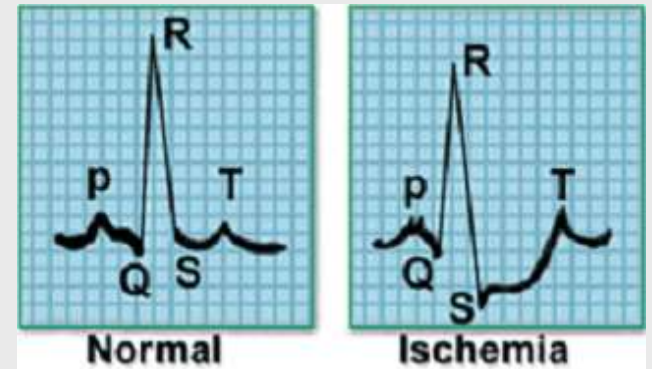
What types of tasks do we want to accomplish?

Similarity (*podobnost*) is a key point

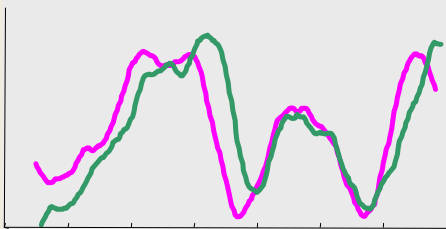
Clustering



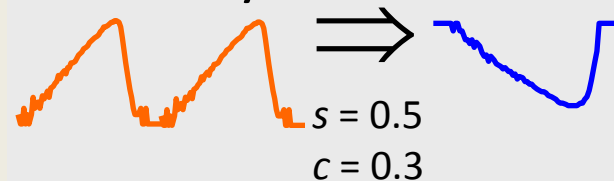
Classification



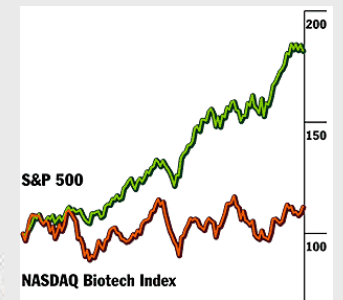
Motif Discovery



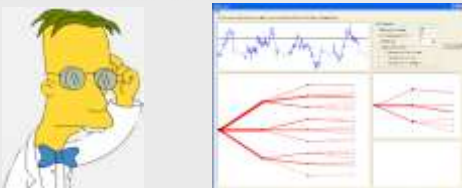
Rule Discovery



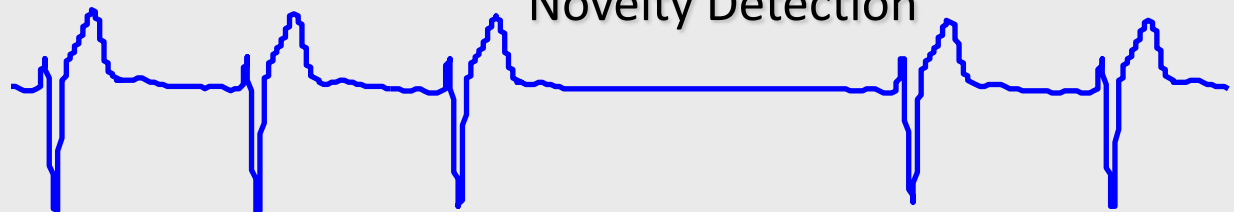
Query by Content



Visualization



Novelty Detection



Why is Working With Time Series so Difficult? Part I

1. Huge size of data → requirements on very efficient representation and on the used algorithms (so that the external memory does not have to be accessed too often).

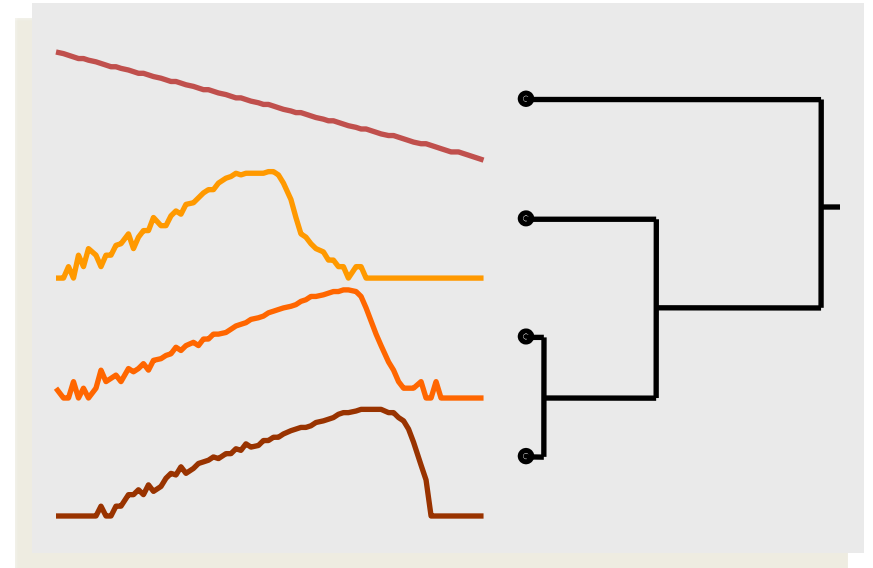
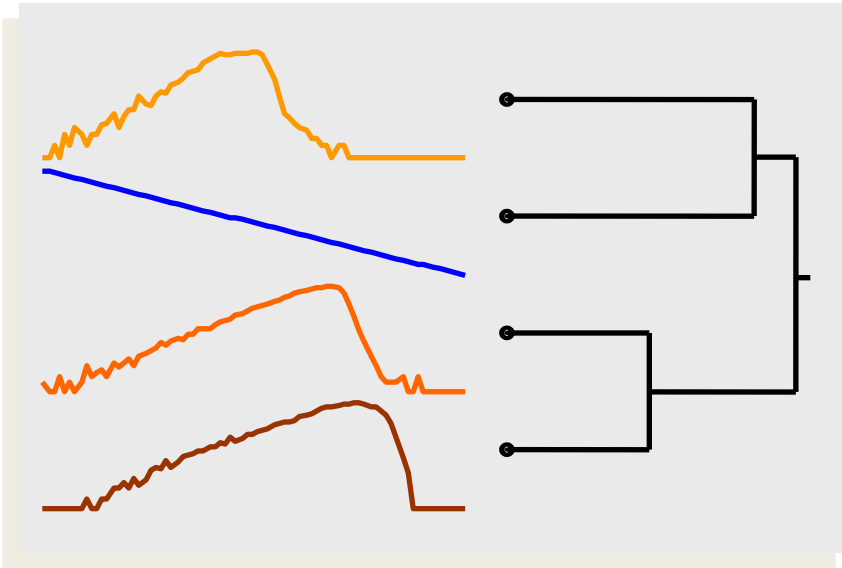
- 1 Hour of EKG data: 1 Gigabyte.
- Typical Weblog: 5 Gigabytes per week.
- Space Shuttle Database: 200 Gigabytes and growing.

2. Problems resulting from the need to merge data from various sources:

- Different format.
- Nonequal sampling frequencies
- Noise, lacking values,

Why is Working With Time Series so Difficult? Part II

Answer: We are dealing with subjectivity



The definition of similarity depends on the user, the domain and the task at hand. We need to be able to handle this subjectivity.

Defining Distance Measures

Definition: Let O_1 and O_2 be two objects from the universe of possible objects. The distance (dissimilarity) is denoted by $D(O_1, O_2)$

What properties are desirable in a distance measure?

- $D(A, B) = D(B, A)$

Symmetry

- $D(A, A) = 0$

Constancy

- $D(A, B) = 0$ iff $A = B$

Positivity

- $D(A, B) \leq D(A, C) + D(B, C)$

Triangular Inequality



Why is the Triangular Inequality so Important?

Virtually all techniques to index data require the triangular inequality to hold.

Why? Let us suppose that given Q we are expected to select among 3 points a , b and c the point with minimal distance to Q .

I find a and calculate that it is 2 units from Q , it becomes my *best-so-far*. I find b and calculate that it is **7.81** units away from Q .

I don't have to calculate the distance from Q to c !

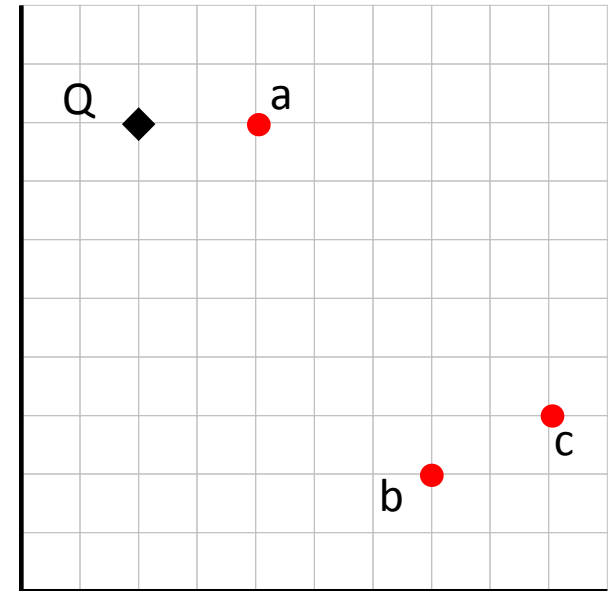
I know $D(Q, \mathbf{b}) \leq D(Q, \mathbf{c}) + D(\mathbf{b}, \mathbf{c})$

$$D(Q, \mathbf{b}) - D(\mathbf{b}, \mathbf{c}) \leq D(Q, \mathbf{c})$$

$$7.81 - 2.30 \leq D(Q, \mathbf{c})$$

$$5.51 \leq D(Q, \mathbf{c})$$

So I know that c is at least 5.51 units away, but my *best-so-far* is only 2 units away.



	a	b	c
a		6.70	7.07
b			2.30
c			

Euclidean Distance Metric

Given two time series:

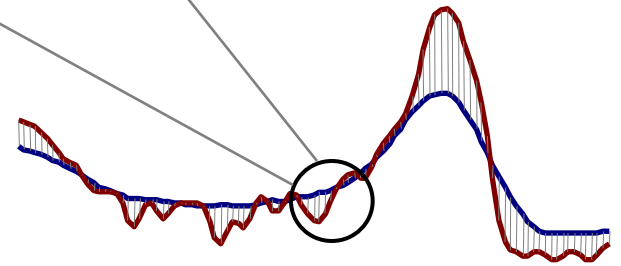
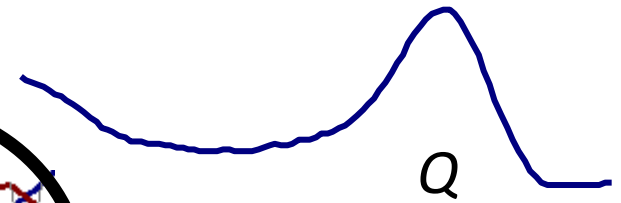
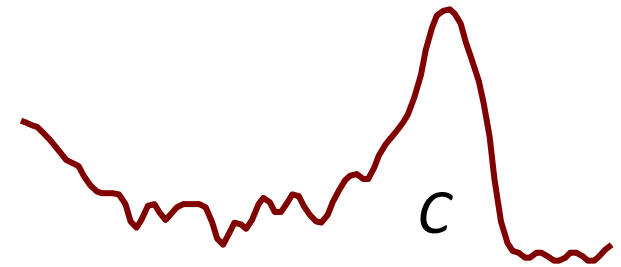
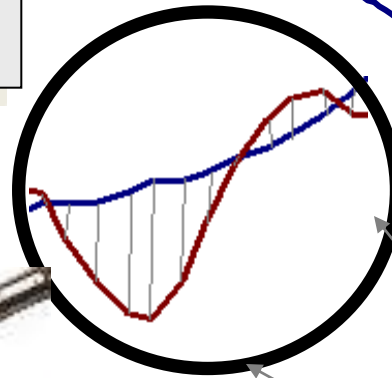
$$Q = q_1 \dots q_n$$

$$C = c_1 \dots c_n$$

$$D(Q, C) \equiv \sqrt{\sum_{i=1}^n (q_i - c_i)^2}$$



About 80% of published work in data mining uses Euclidean distance

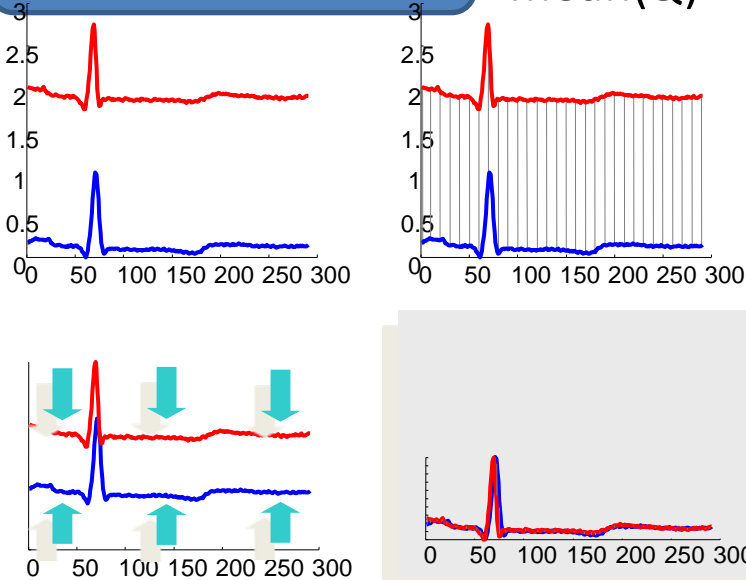


$D(Q, C)$

Preprocessing and Linear Transformations

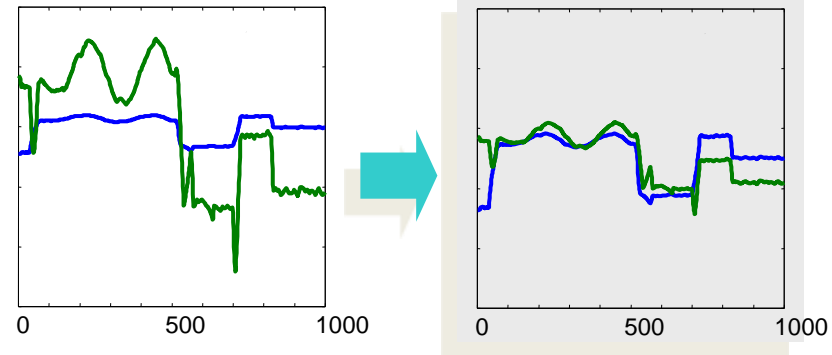
TI: Offset Translation

$$Q = Q - \text{mean}(Q)$$



TII: Amplitude Scaling

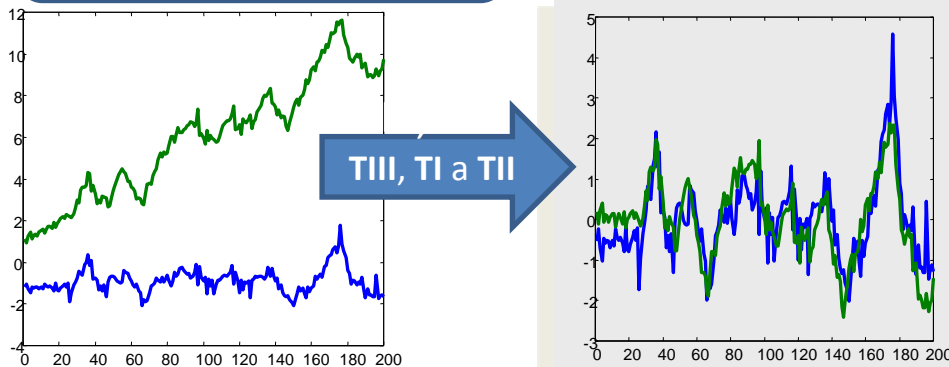
$$Q = (Q - \text{mean}(Q)) / \text{std}(Q)$$



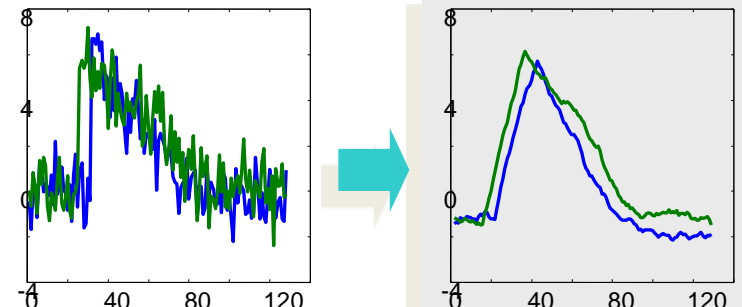
TIV: Noise

TIII: Linear Trend

The original signal s_1 is approximated by a line l_1 : the new signal is their difference ($s_1 - l_1$).

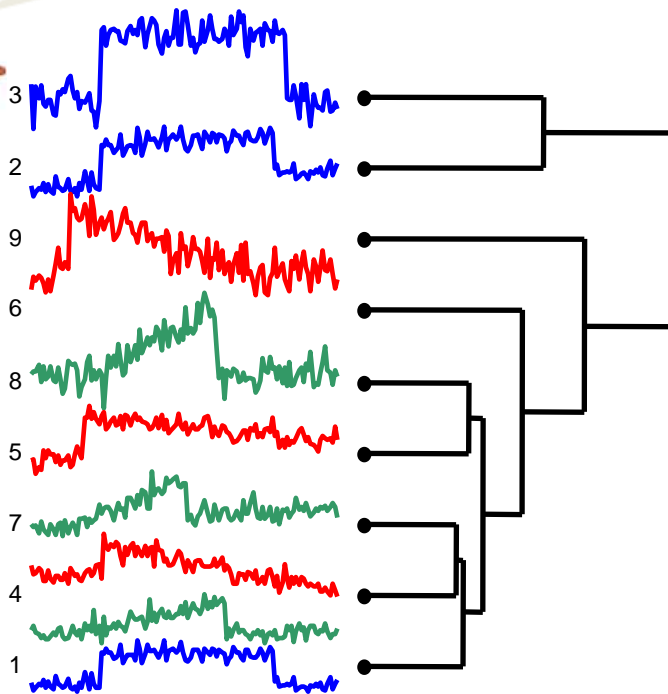


Smoothing: Each value of the signal is replaced by the average of the values in its close neighborhood.

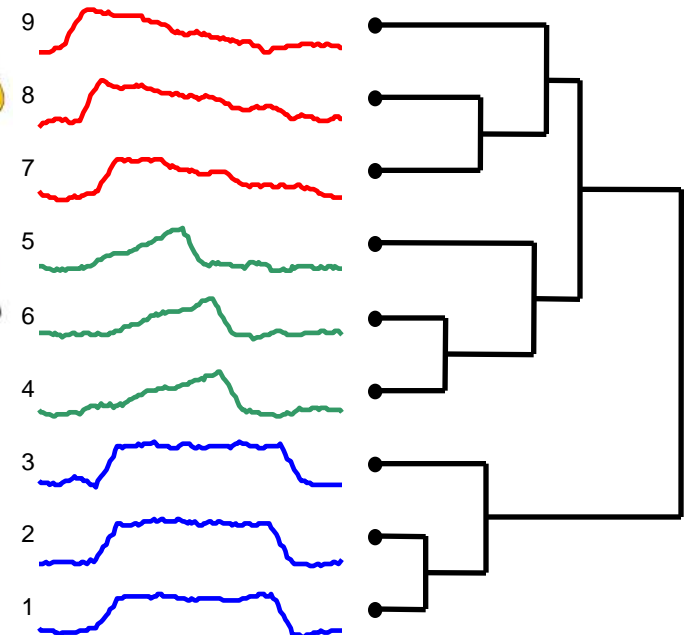


A Quick Experiment to Demonstrate the Utility of Preprocessing the Data

Clustered using Euclidean distance on the raw data.

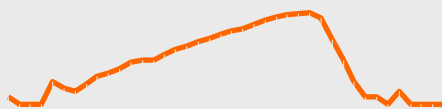
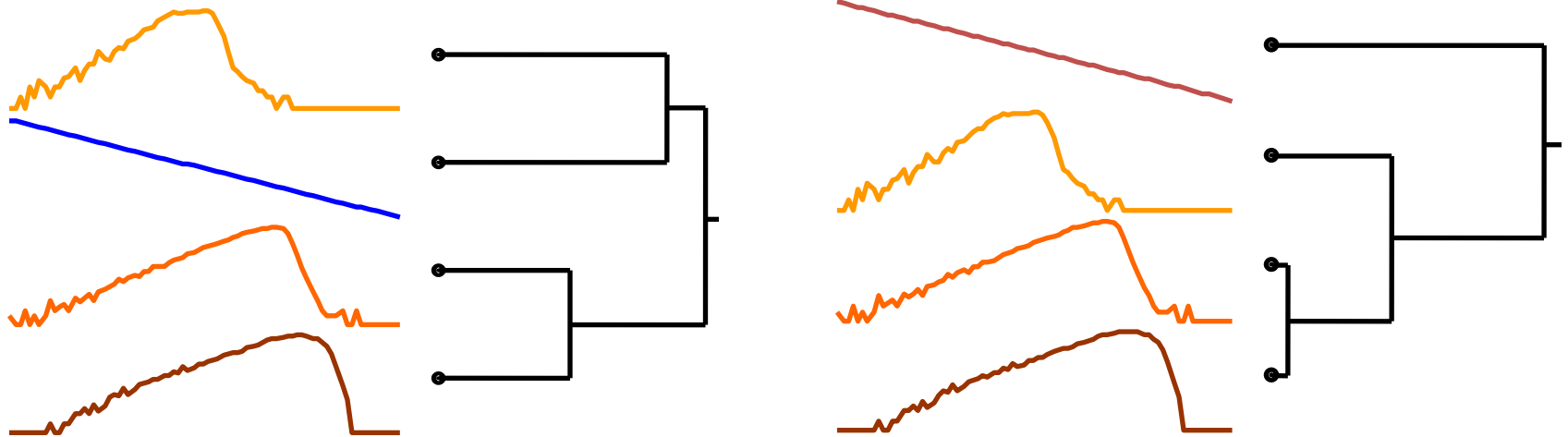


Clustered using Euclidean distance, after removing noise, linear trend, offset translation and amplitude scaling



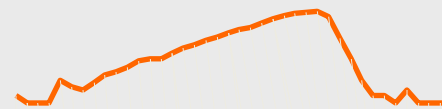
Dynamic Time Warping

Dynamické borcení času



Fixed Time Axis

Sequences are aligned "one to one".

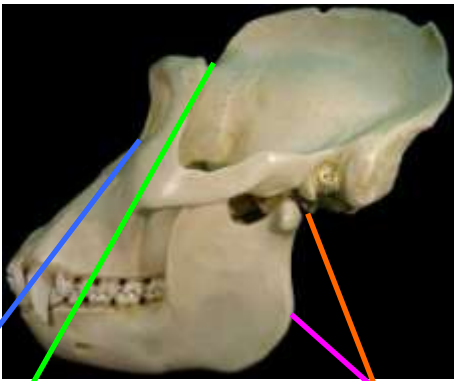


"Warped" Time Axis

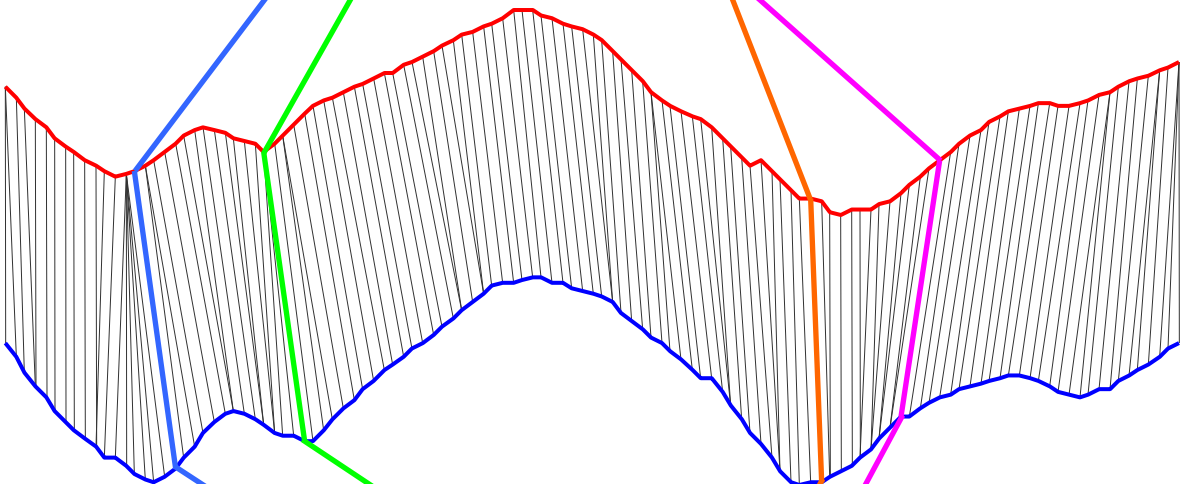
Nonlinear alignments are possible.

Note: We will first see the utility of DTW, then see how it is calculated.

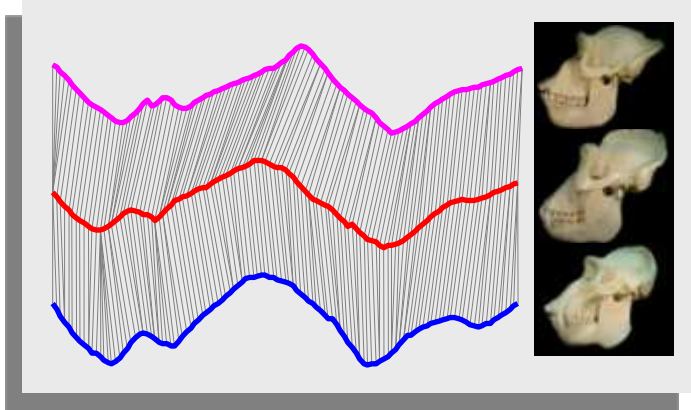
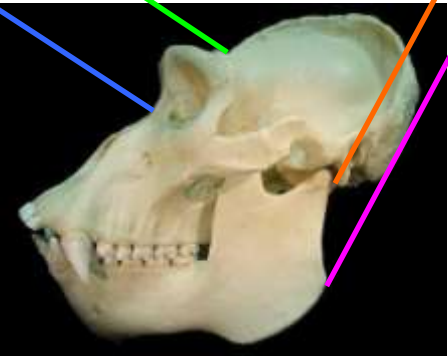
Lowland Gorilla
Gorilla gorilla graueri



DTW is needed
for most natural
objects...

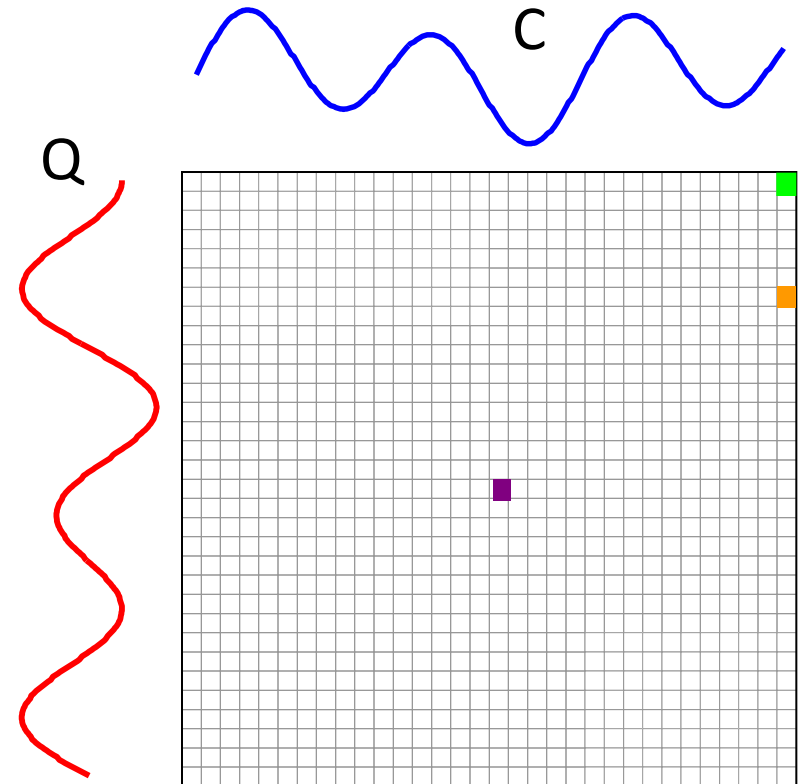
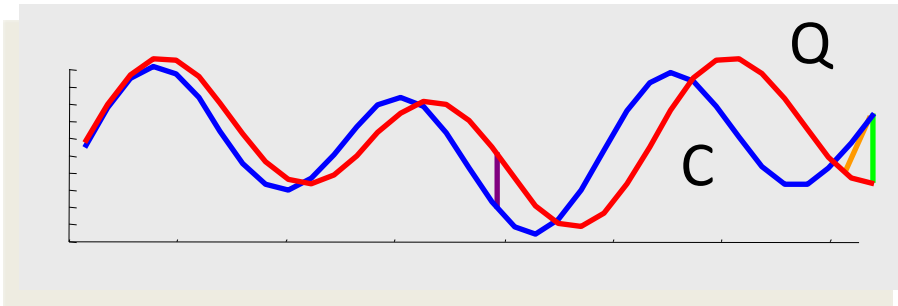


Mountain Gorilla
Gorilla gorilla beringei



How is DTW Calculated? I

We create a matrix the size of $|Q|$ by $|C|$, then fill it in with the distance between every pair of points in our two time series.



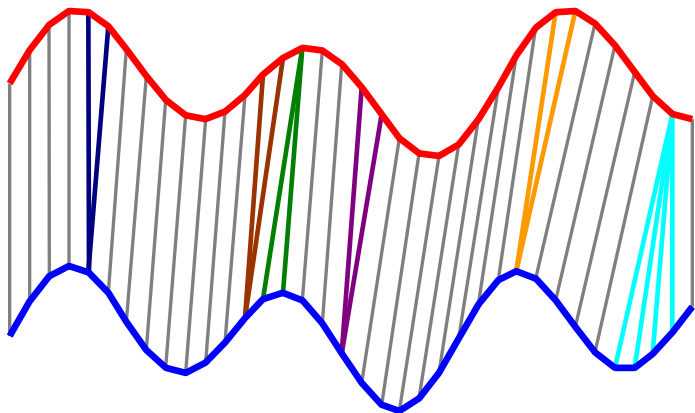
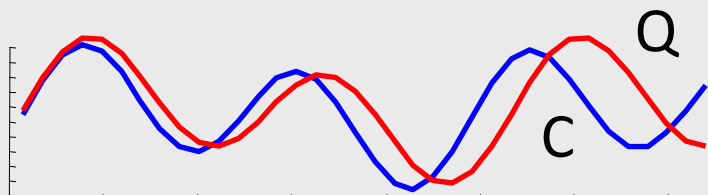
Estimated complexity (number of all possible WPs) in a matrix $n \times n$?

Upper bound is 3^n

How is DTW Calculated? II

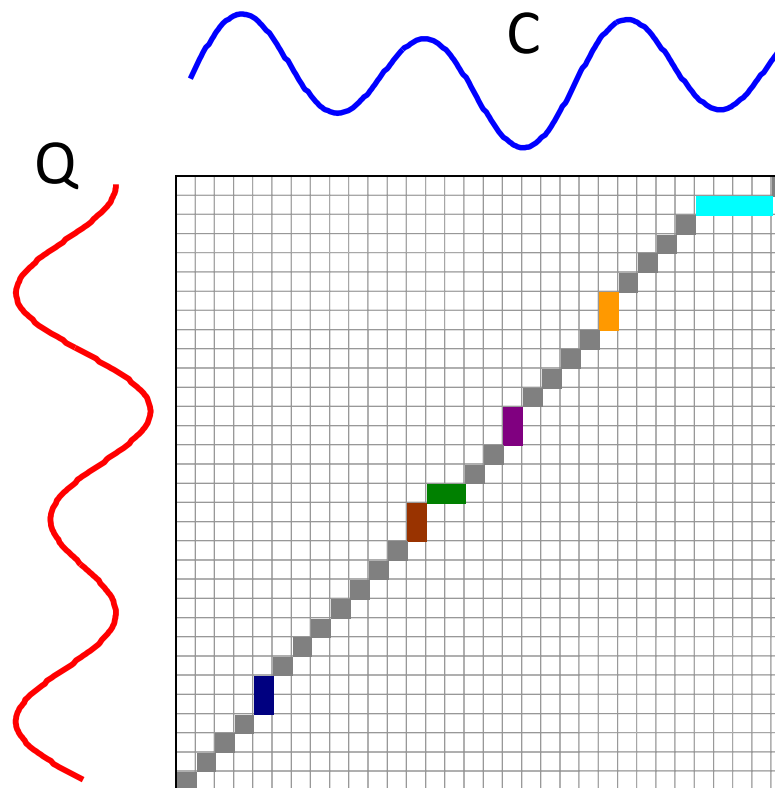
Every possible warping between two time series, is a path through the matrix. We want the best one...

$$DTW(Q, C) = \min \left\{ \sqrt{\sum_{k=1}^K w_k} / K \right\}$$



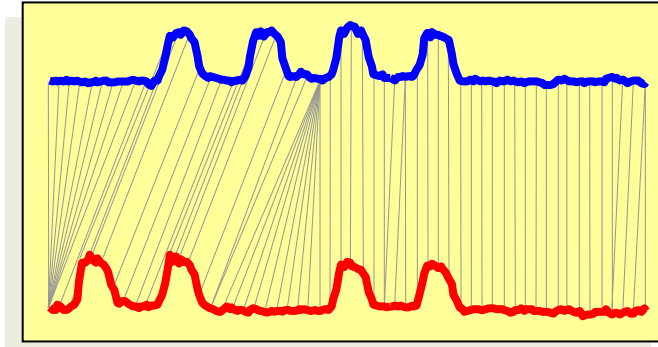
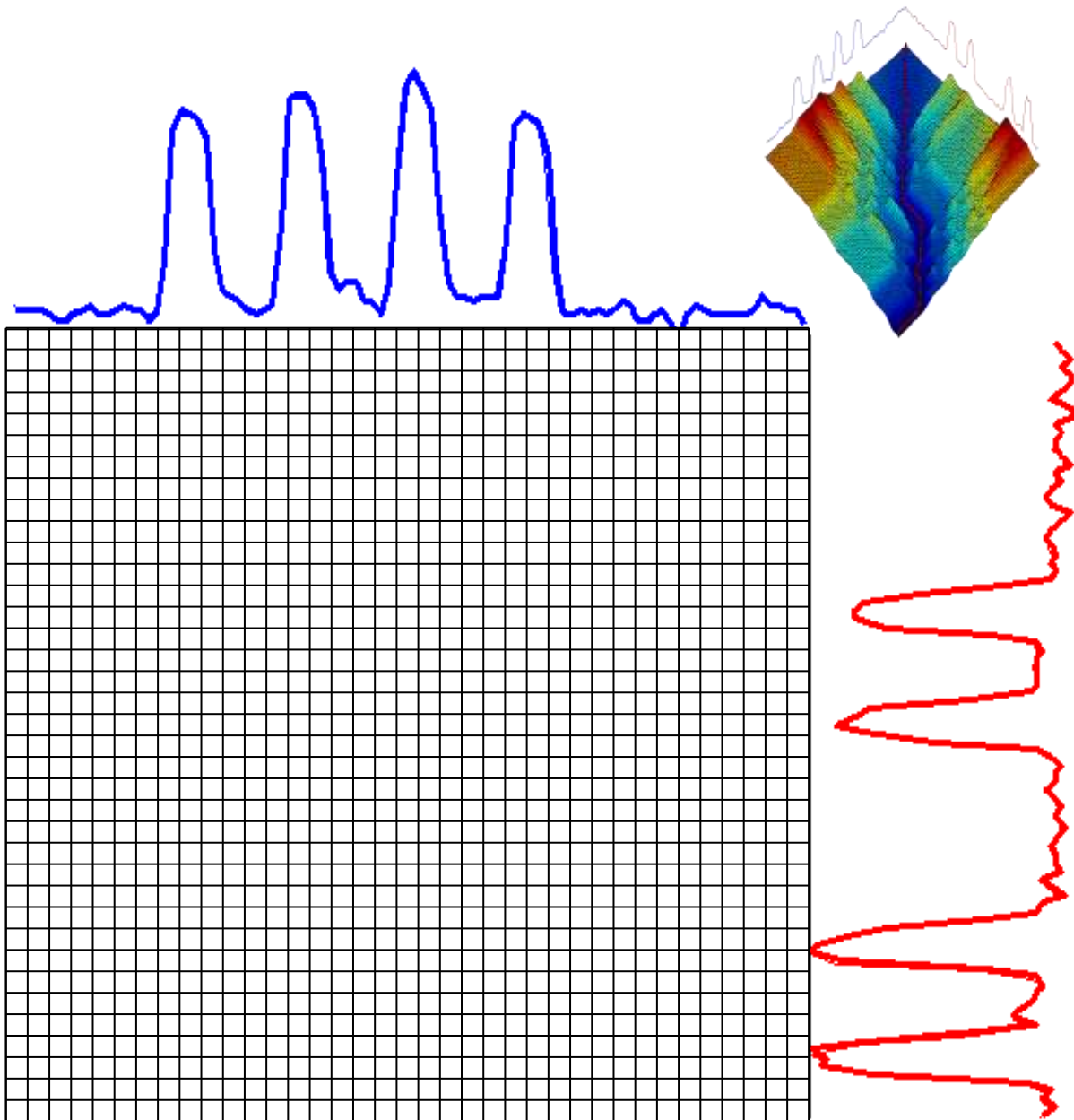
This recursive function gives us the minimum cost path

$$\gamma(i, j) = d(q_i, c_j) + \min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\}$$



Warping path w

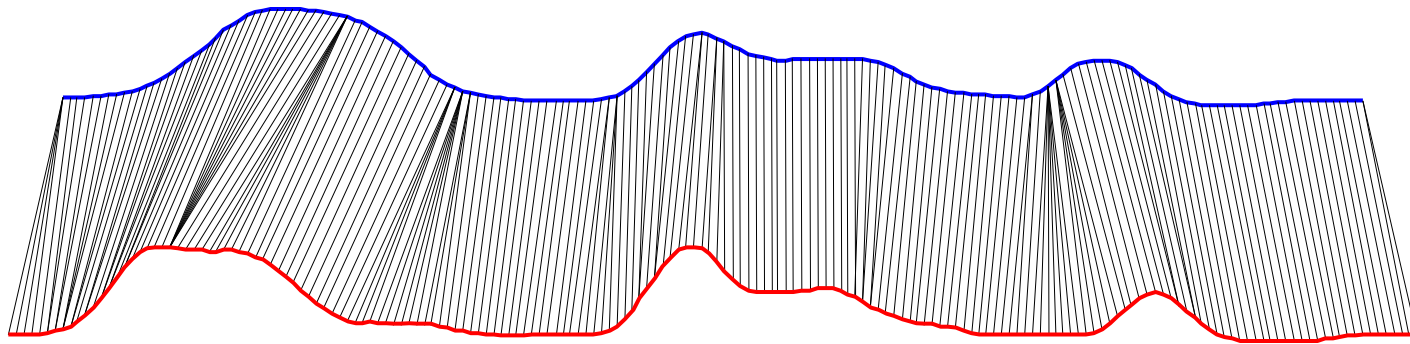
Let us visualize the cumulative matrix on a real world problem I



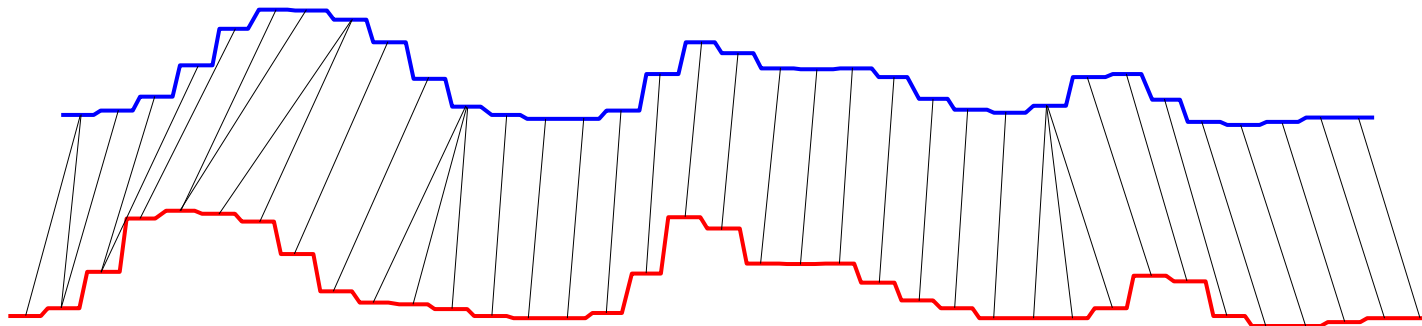
This example shows 2 one-week periods from the power demand time series.

Note that although they both describe 4-day work weeks, the blue sequence had Monday as a holiday, and the red sequence had Wednesday as a holiday.

Are **Fast Approximations** to Dynamic Time Warp Distance Useful?



1.03 sec



0.07 sec

... there is strong visual evidence to suggests it works well

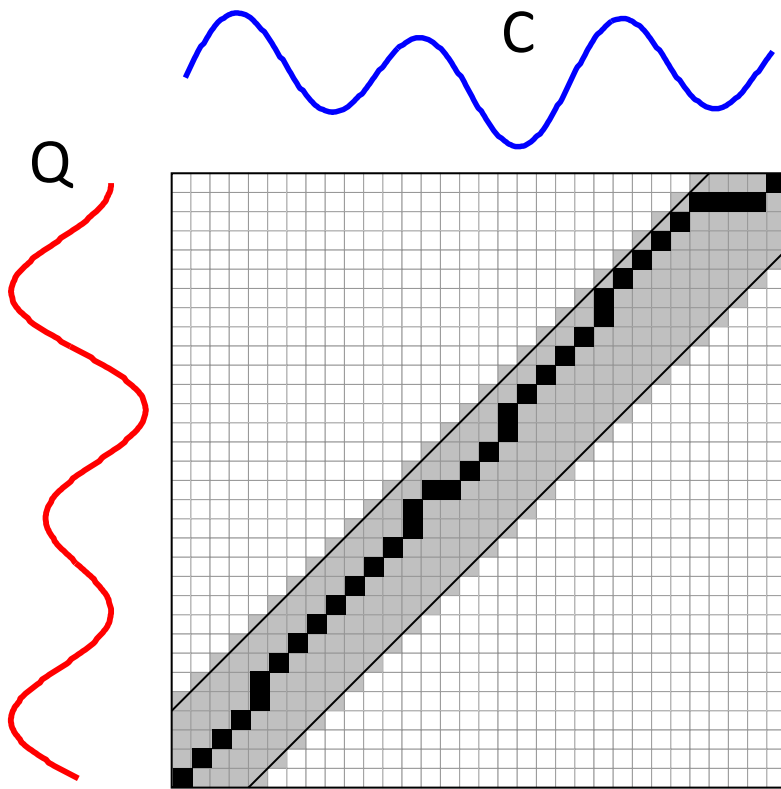
There is good experimental evidence for the utility of the approach on clustering, classification, etc



Global Constraints

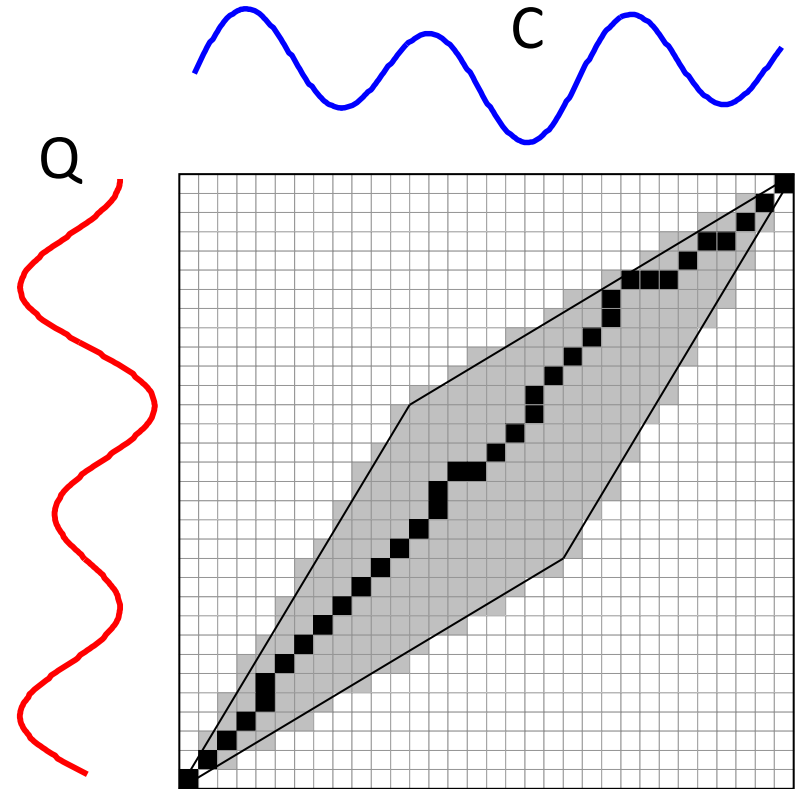
- Slightly speed up the calculations
- Prevent pathological warpings

Sakoe-Chiba Band



Warping Window
Size

Itakura Parallelogram



How to speed up calculation of distance?

Lower Bounding

We can speed up similarity search under DTW by using a **lower bounding function**

Algorithm Lower_Bounding_Sequential_Scan(Q)

```
1.  best_so_far = infinity;
2.  for all sequences in database
3.    LB_dist = lower_bound_distance( Ci, Q);
4.    if LB_dist < best_so_far
5.      true_dist = DTW( Ci, Q);
6.      if true_dist < best_so_far
7.        best_so_far = true_dist;
8.        index_of_best_match = i;
9.      endif
10.   endif
11. endfor
```

Try to use a cheap lower bounding calculation as often as possible.



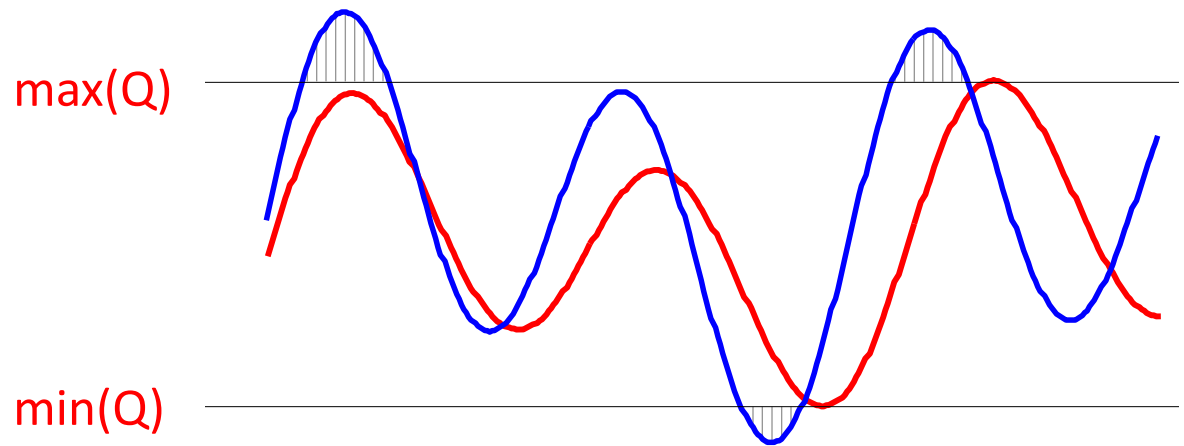
Only do the expensive, full calculations when it is absolutely necessary



Lower Bound of Y_i



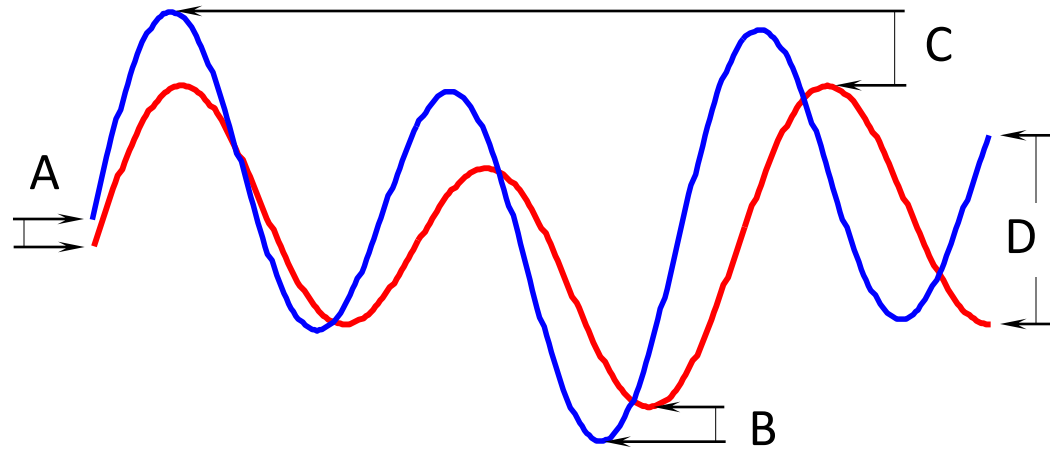
LB_ Y_i



The sum of the squared length of gray lines represent the minimum the corresponding points contribution to the overall DTW distance, and thus can be returned as the lower bounding measure

Yi, B, Jagadish, H & Faloutsos, C.
Efficient retrieval of similar time sequences under time warping.
ICDE 98, pp 23-27.

Lower Bound of Kim

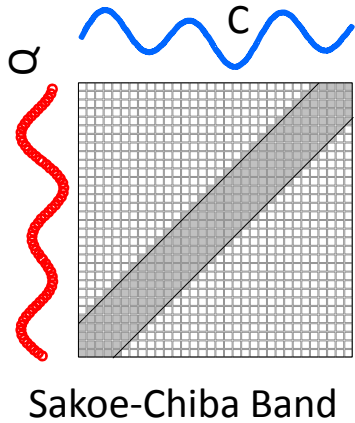


LB_Kim

Kim, S, Park, S, & Chu, W. *An index-based approach for similarity search supporting time warping in large sequence databases*. ICDE 01, pp 607-614

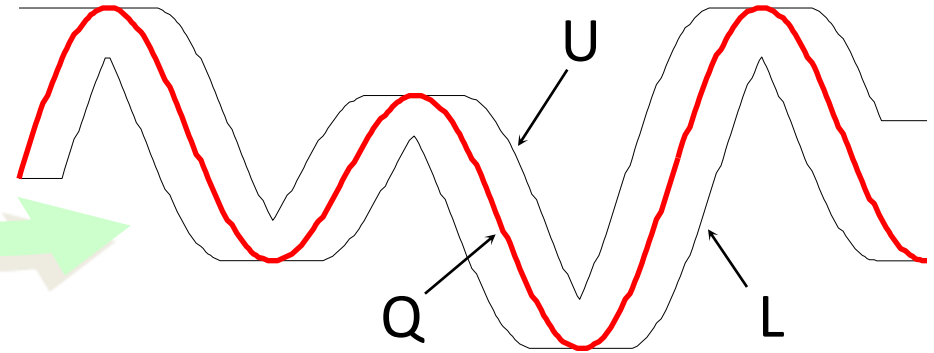
The squared difference between the two sequence's first (A), last (D), **minimum** (B) and **maximum points** (C) is returned as the lower bound

Lower Bound of Keogh



$$U_i = \max(q_{i-r} : q_{i+r})$$

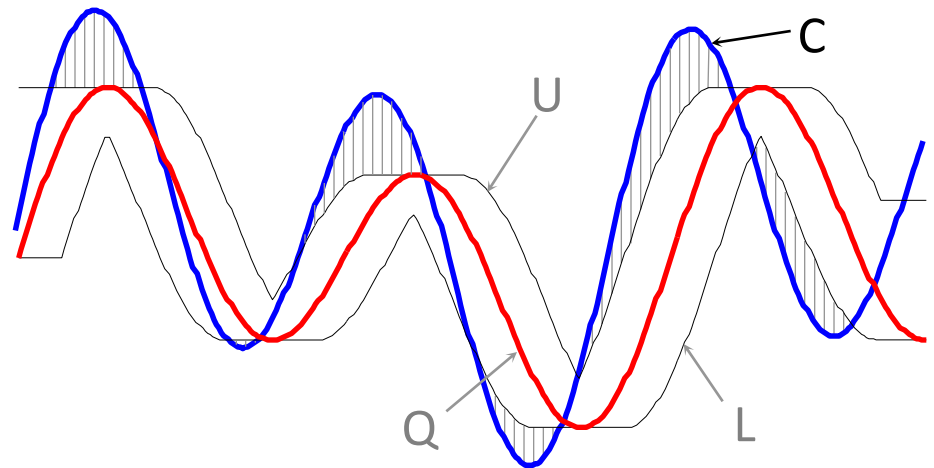
$$L_i = \min(q_{i-r} : q_{i+r})$$



LB_Keogh

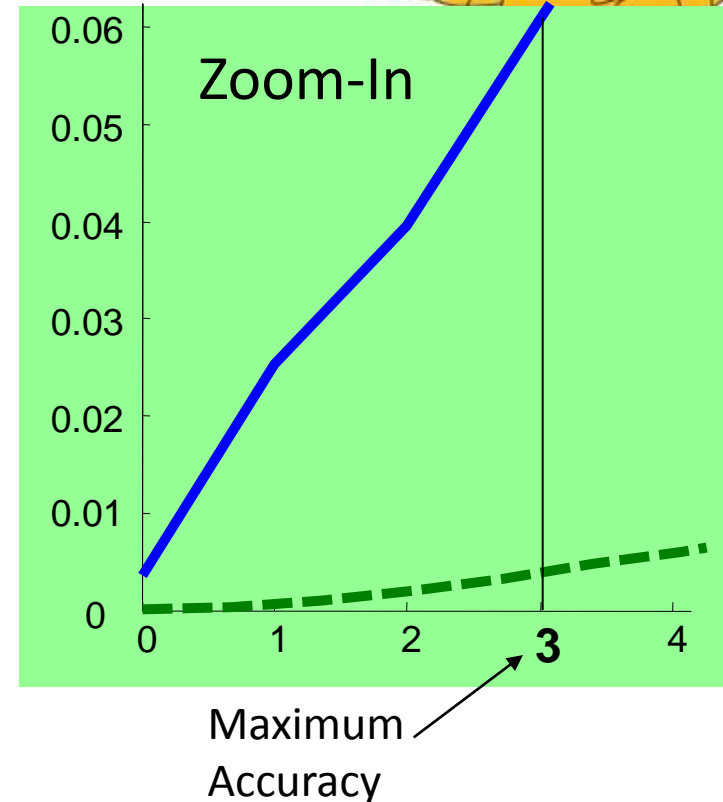
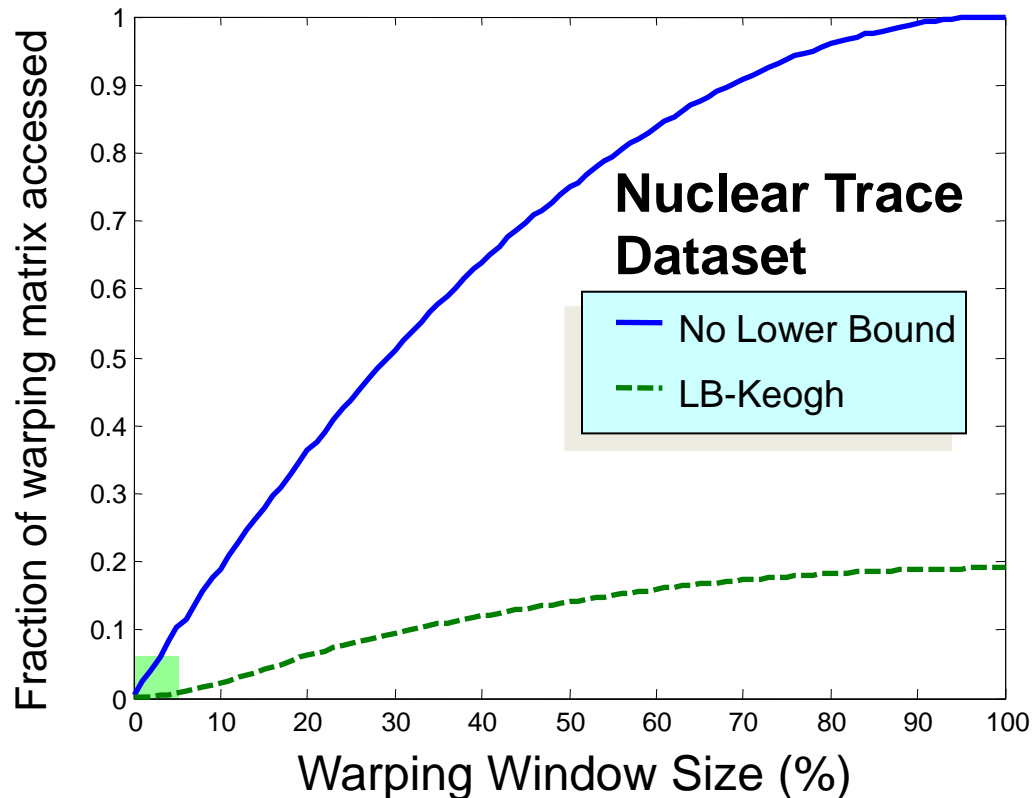
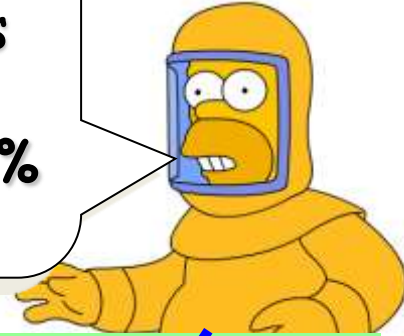
Envelope-Based Lower Bound

$$LB_Keogh(Q, C) = \sum_{i=1}^n \begin{cases} (q_i - U_i)^2 & \text{if } q_i > U_i \\ (q_i - L_i)^2 & \text{if } q_i < L_i \\ 0 & \text{otherwise} \end{cases}$$



How Useful are Lower Bounds?

This plot tells us that although DTW is $O(n^2)$, after we set the warping window for maximum accuracy for this problem, we only have to do 6% of the work, and if we use the LB_Keogh lower bound, we only have to do **0.3%** of the work!



Frequent myths about DTW ...

- *“DTW incurs a heavy CPU cost”¹*
- *“DTW is limited to only small time series datasets”²*
- *“(DTW) quadratic cost makes its application on databases of long time series very expensive”³*
- *“(DTW suffers from) serious performance degradation in large databases”⁴*

This is simply not true!

...DTW is linear for data mining problems!

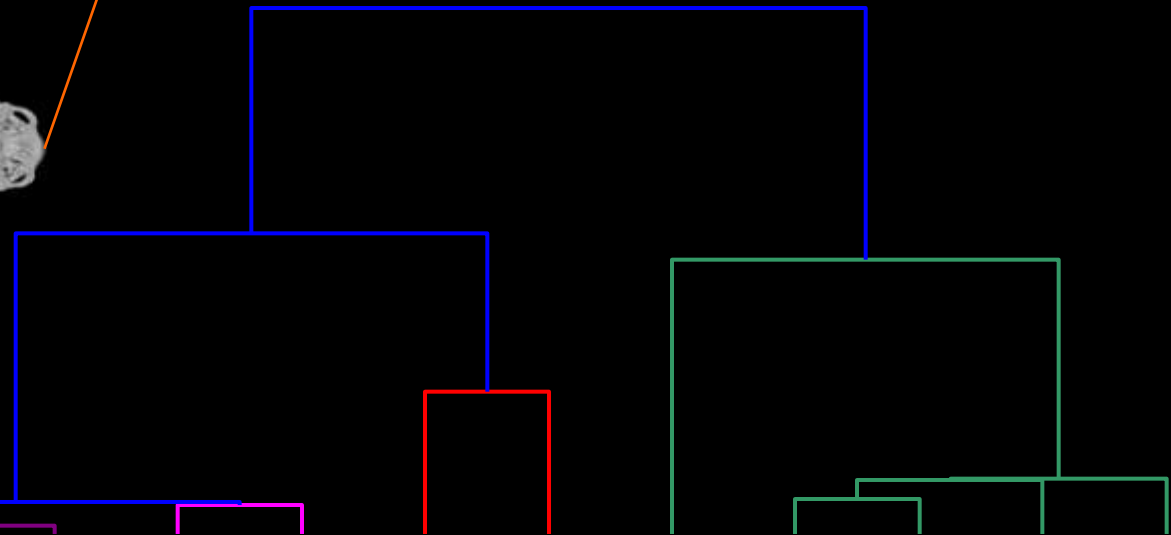
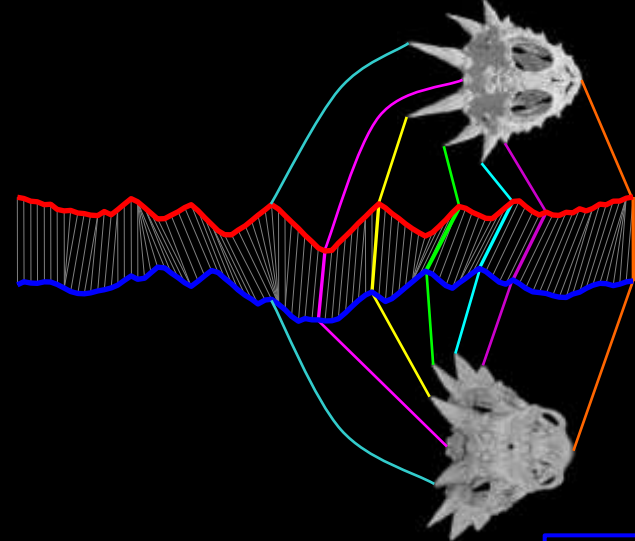




LB_Keogh can be used to index shapes with rotation invariance

Success Sto

IIII



- Alligator mississippiensis*
- Caiman crocodilus*
- Crocodylus cataphractus*
- Tomistoma schlegelii*
- Cricosaura typica*
- Xantusia vigilis*
- Elseya dentata*
- Glyptemys muhlenbergii*
- Phrynosoma braconnieri*
- Phrynosoma ditmarsii*
- Phrynosoma taurus*
- Phrynosoma douglassii*
- Phrynosoma hernandesi*

Codylidae

Alligatoridae

Amphisbaenia

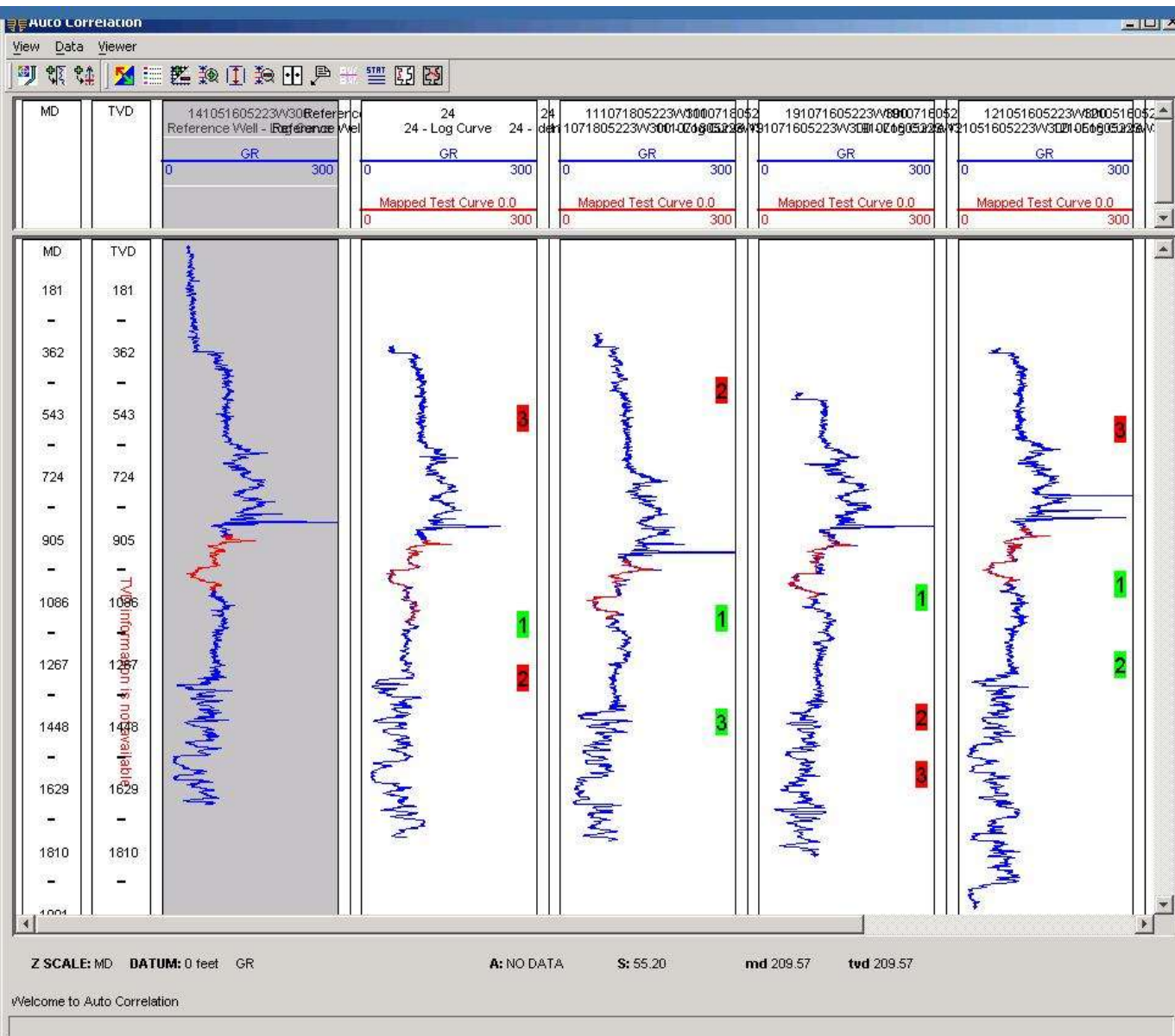
Chelonia

Iguania

Success Story

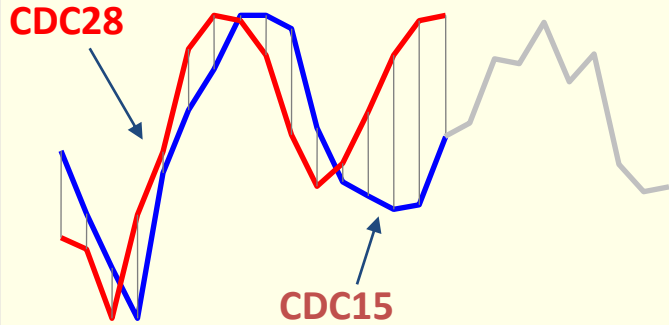
The lower bounding technique is being used by ChevronTexaco for comparing seismic data

Thanks of Steve Zoraster for the figure

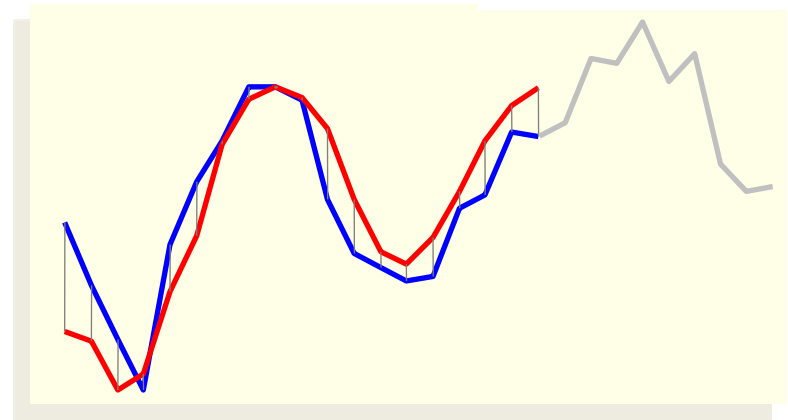
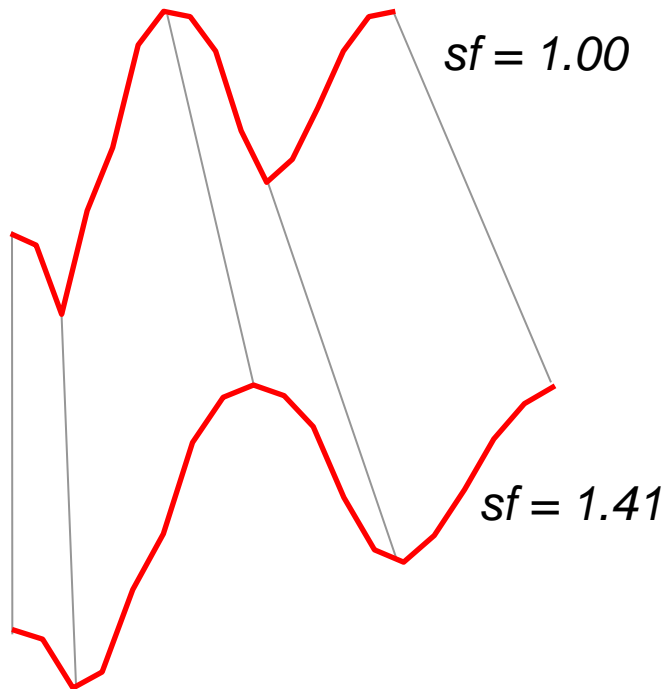


Uniform Scaling I

Two genes that are known to be functionally related...

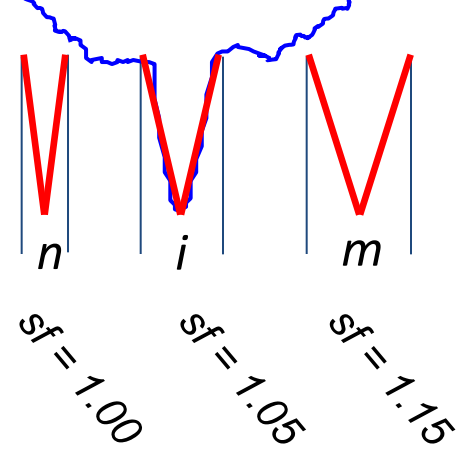


Sometimes global or *uniform scaling* is as important as DTW





Here is some notation, the shortest scaling we consider is length n , and the largest is length m . The *scaling factor* (sf) is the ratio i/n , $n \leq i \leq m$



Algorithm: `Test_All_Scalings(Q,C)`

```
best_match_val      = inf;
best_scaling_factor = null;
for p = n to m
  QP = rescale(Q,p);
  distance = squared_Euclidean_distance(QP, C[1..p])
  if distance < best_match_val
    best_match_val = distance;
    best_scaling_factor = p/n;
  end;
end;
return(best_match_val, best_scaling_factor)
```


Here is the code to `Test_All_Scalings`, the time complexity is only $O((m-n) * n)$, but we may have to do this many times...




Lower Bounding Revisited!

We can speed up similarity search under uniform scaling by using a lower bounding function, just like we did for DTW.

```
Algorithm: Lower_Bounding_Sequential_Scan(Q,C)
overall_best_time_series = null;
overall_best_match_val = inf;
for i = 1 to number_of_time_series_in_(C)
  if lower_bound_distance(Q,Ci) < overall_best_match_val
    [dist, scale] = Test_All_Scalings(Q,Ci)
    if dist < overall_best_match_val
      overall_best_time_series = i;
      overall_best_match_val = dist;
    end;
  end;
end;
```



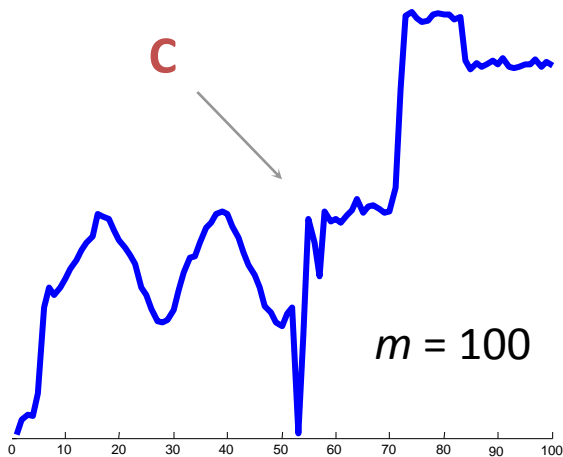
You have already seen this idea for DTW!



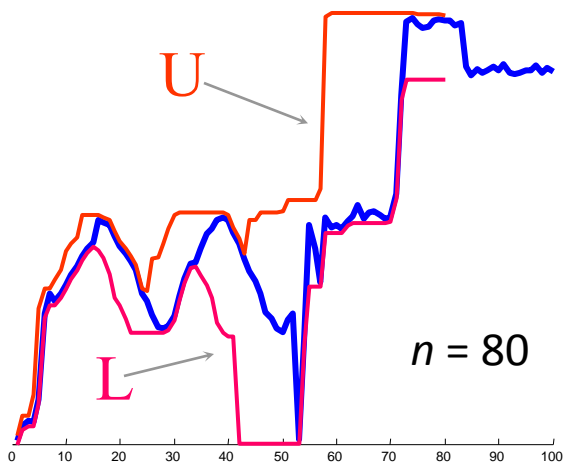
But is there a lower bound for uniform scaling?

Assume that you have a database of time series C_i , all of length 100.

You have a query Q , of length 80, and you want to find the best match in the database under any scaling of Q , from 80 to 100.



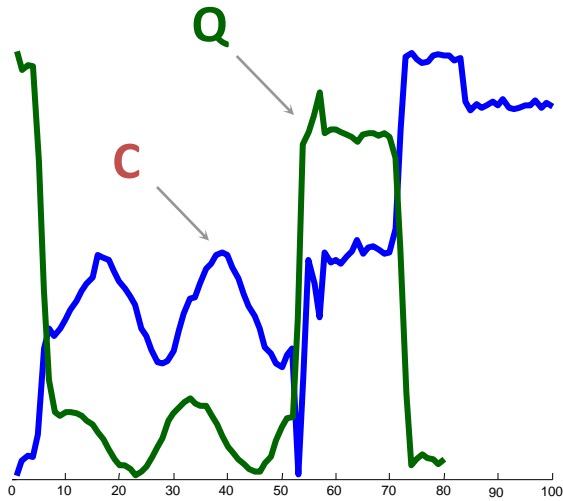
We can build envelopes around all candidates time series C_i , in our database, just like we did for DTW, except the definition of the envelopes is different.



$$U_i = \max(c_{\lfloor (i-1)*m/n \rfloor + 1}, \dots, c_{\lfloor i*m/n \rfloor})$$

$$L_i = \min(c_{\lfloor (i-1)*m/n \rfloor + 1}, \dots, c_{\lfloor i*m/n \rfloor})$$





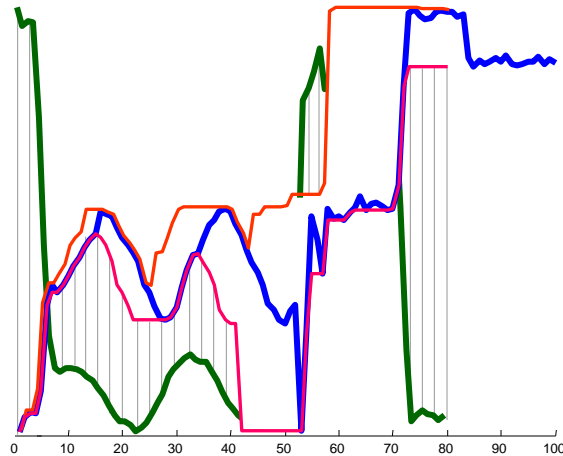
Once the envelopes have been built, we can lower bound **Test_All_Scalings**.

What's more, the lower bound is one we have already seen!



LB_Keogh

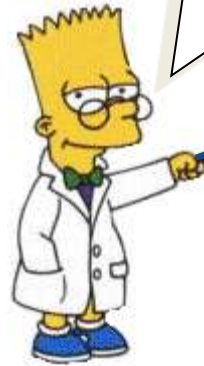
Envelope-Based Lower Bound



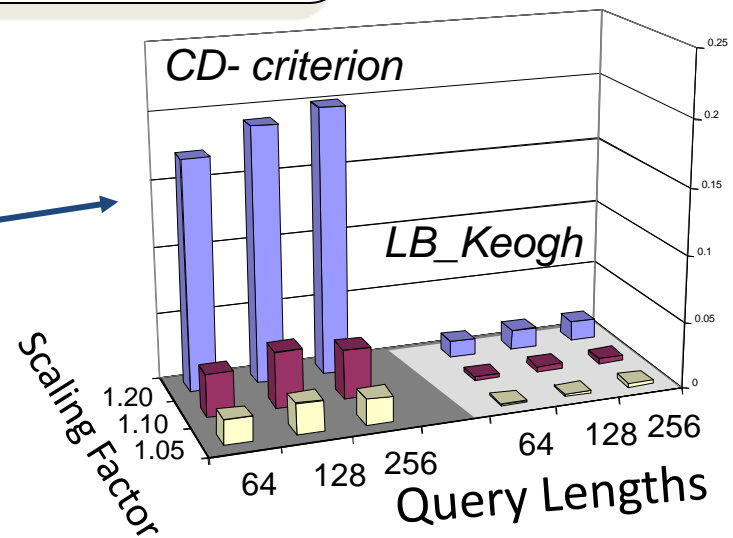
$$LB_Keogh(Q, C) = \sum_{i=1}^n \begin{cases} (q_i - U_i)^2 & \text{if } q_i > U_i \\ (q_i - L_i)^2 & \text{if } q_i < L_i \\ 0 & \text{otherwise} \end{cases}$$


An experiment to test the utility of lower bounding uniform scaling, over different scaling factors (Y-axis) and scaling lengths (X-axis). The dataset was a “mixed bag” of 10,000 assorted time series.

This is the time taken by brute force search



CD-criterion is the only other lower bound for uniform scaling





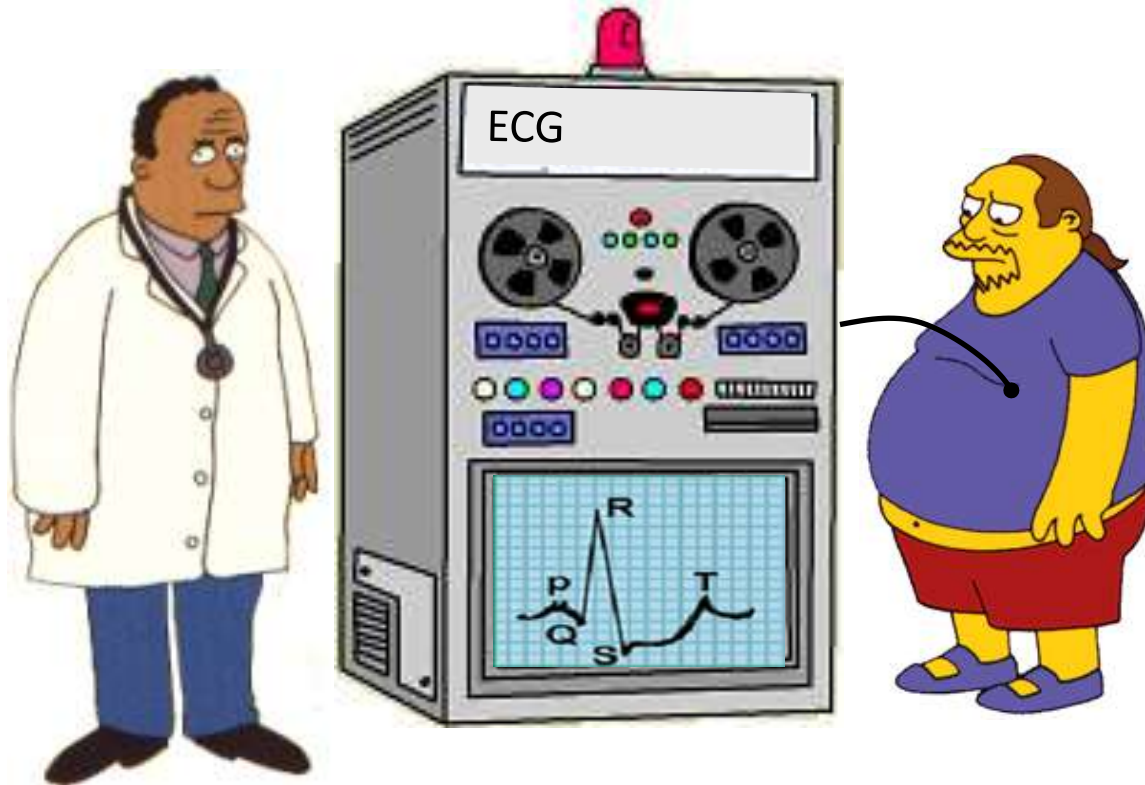
Apart from making DTW tractable for data mining for the first time, *envelope based techniques* also allow...

1. More accurate classification (SDM04)
2. Indexing with uniform scaling (VLDB04)
3. Faster Euclidean indexing (TKDE04)
4. Subsequence matching (IDEAS03)
5. Multivariate time series indexing (SIGKDD03)
6. Rotation invariant indexing (SIGKDD04)
7. DTW on Streaming time series (to appear)
8. Indexing of Images (TPAMI-04, VIS-05)

We strongly feel that envelope based techniques are the best solutions for time series similarity



Motivating example revisited...



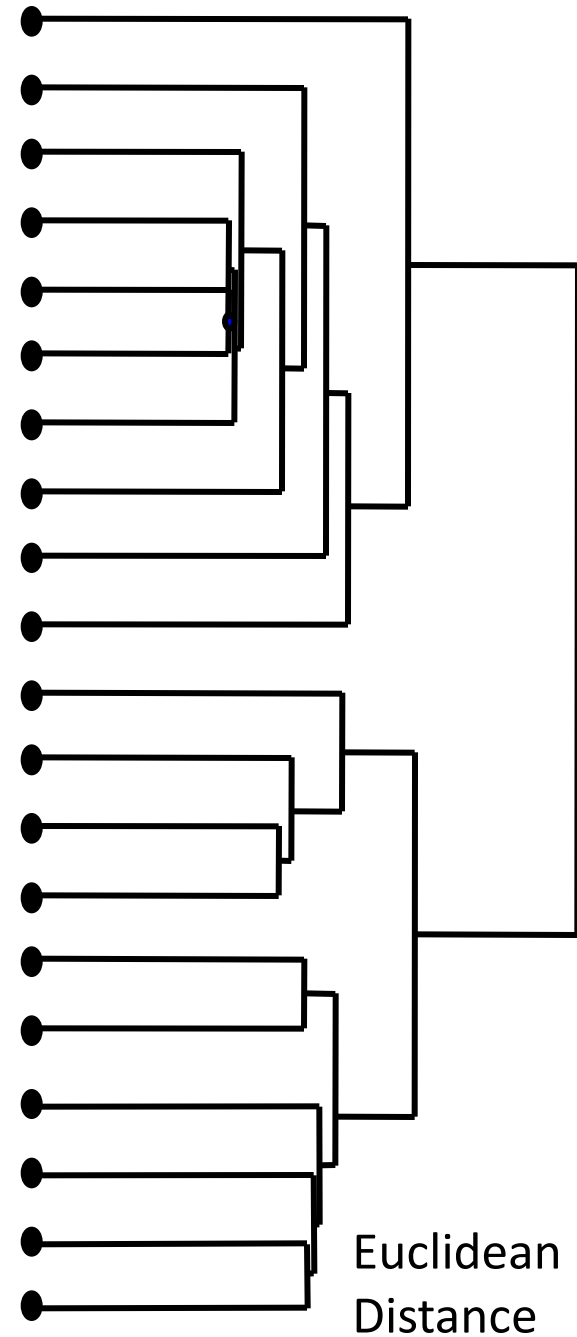
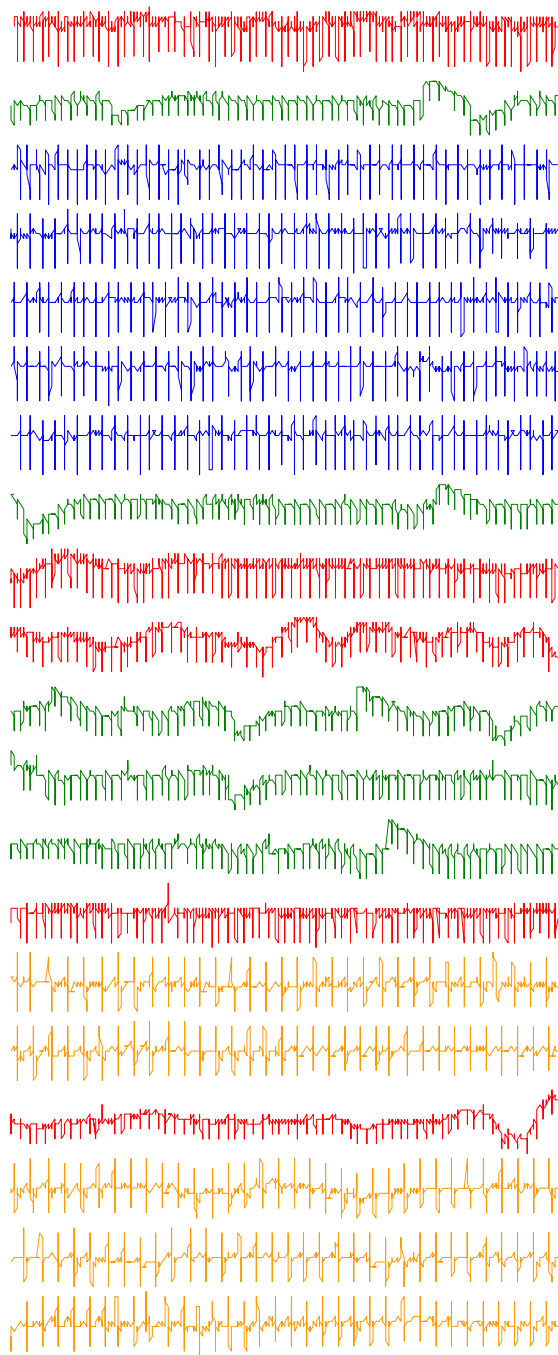
You go to the doctor because of chest pains. Your ECG looks strange...

Your doctor wants to search a database to find **similar** ECGs, in the hope that they will offer clues about your condition...

Two questions:

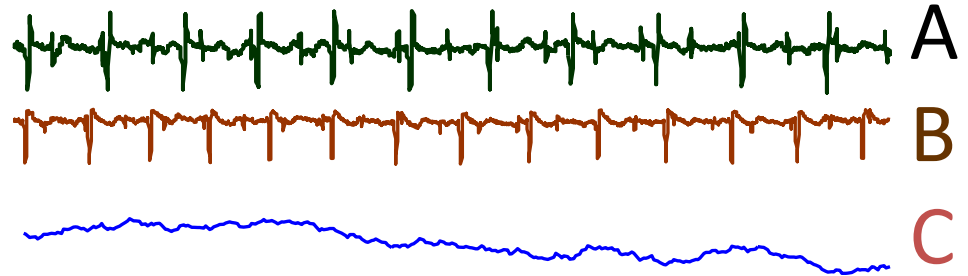
- How do we define similar?
- **How do we search quickly?**

For long time series, *shape* based similarity will give very poor results. We need to measure similarity based on high level *structure*



Structure or Model Based Similarity

The basic idea is to extract *global* features from the time series, create a feature vector, and use these feature vectors to measure similarity and/or classify



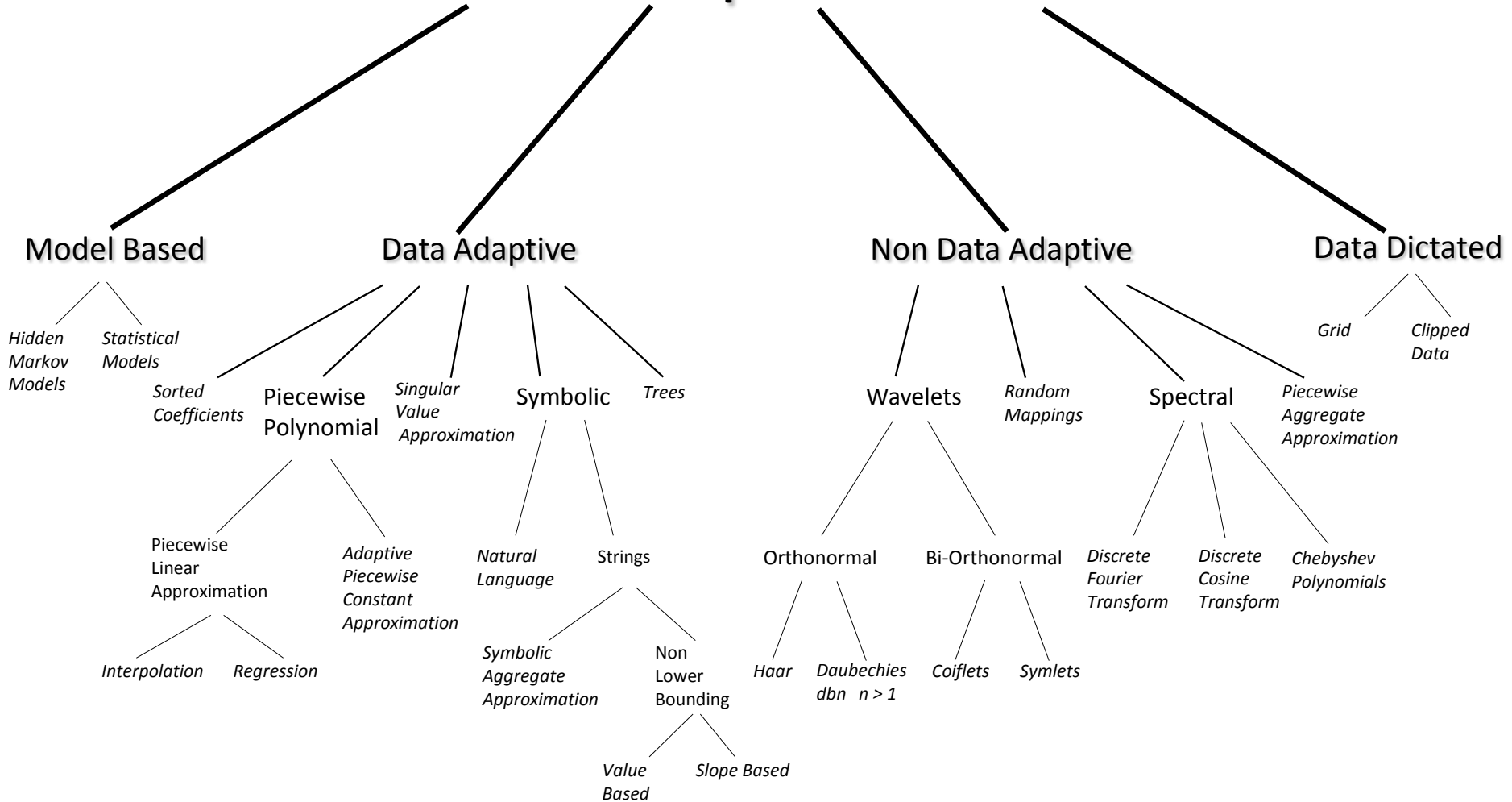
Time Series \ Feature	A	B	C
Max Value	11	12	19
Autocorrelation	0.2	0.3	0.5
Zero Crossings	98	82	13
ARIMA	0.3	0.4	0.1
...

But which

- **features?**
- **distance measure/ learning algorithm?**



Time Series Representations



The Generic Data Mining Algorithm (revisited)

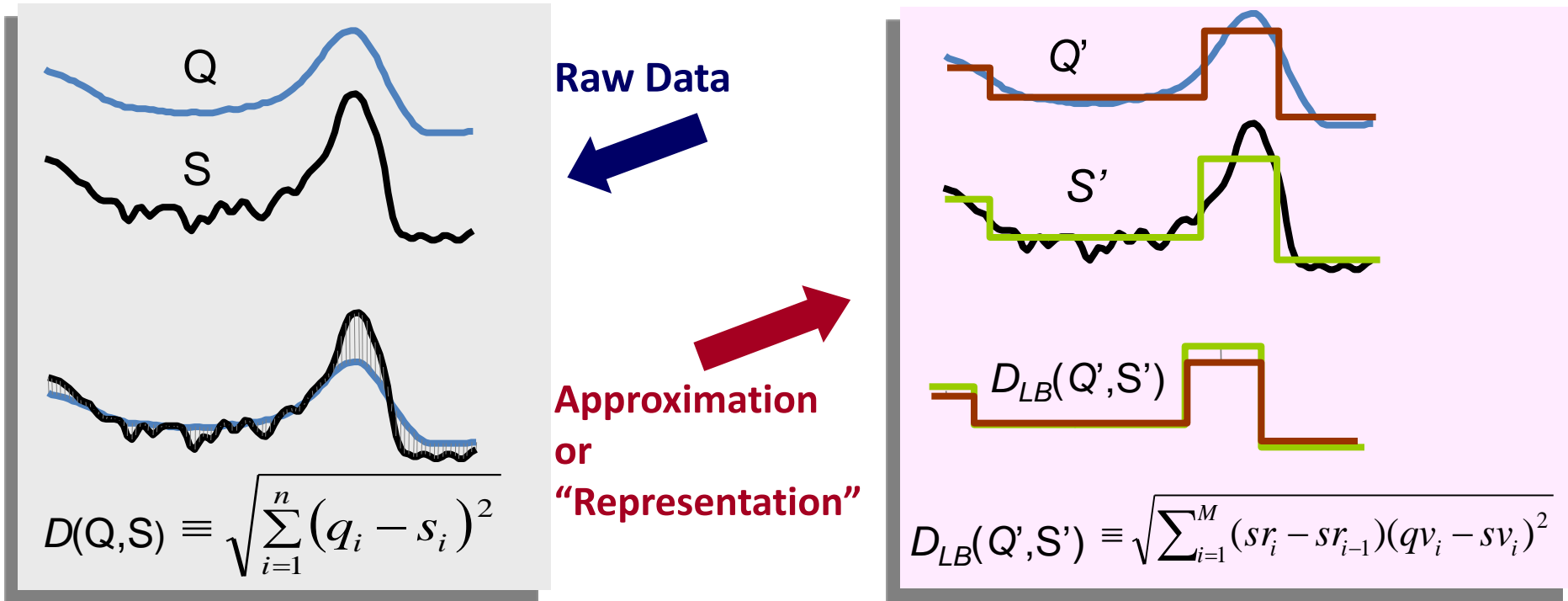
- Create an *approximation* of the data, which will fit in main memory, yet retains the essential features of interest
- **Approximately solve the problem at hand in main memory**
- Make (hopefully very few) accesses to the original data on disk to confirm the solution obtained in Step 2, or to modify the solution so it agrees with the solution we would have obtained on the original data

This only works if the approximation allows lower bounding



What is Lower Bounding?

- Recall that we have seen lower bounding for **distance measures** (DTW and uniform scaling) Lower bounding for **representations** is a similar idea...

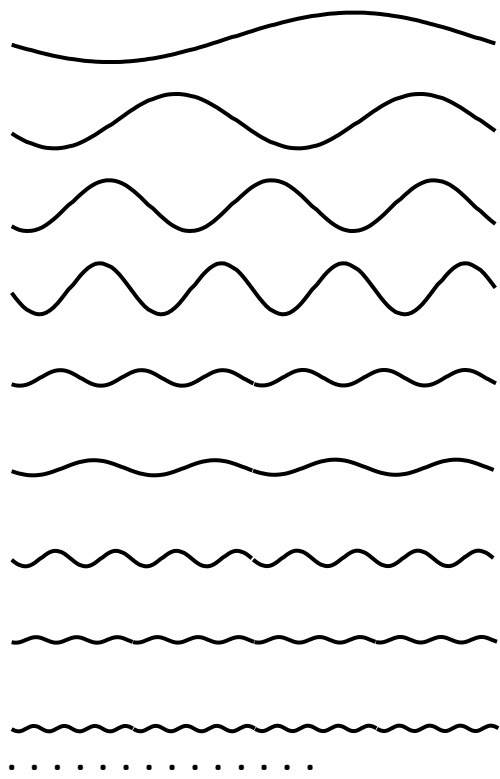
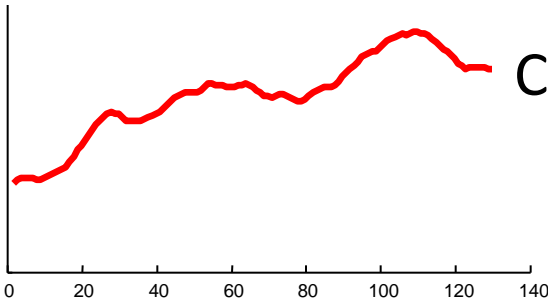


Lower bounding means that for all Q and S, we have:

$$D_{LB}(Q', S') \leq D(Q, S)$$



An Example of a Dimensionality Reduction Technique II



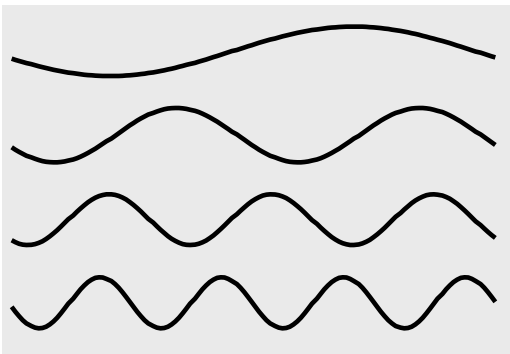
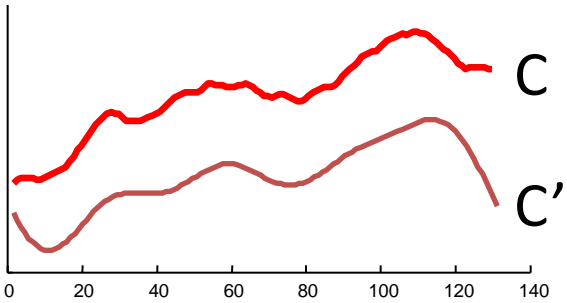
Raw Data	Fourier Coefficients
0.4995	1.5698
0.5264	<u>1.0485</u>
0.5523	0.7160
0.5761	<u>0.8406</u>
0.5973	0.3709
0.6153	<u>0.4670</u>
0.6301	0.2667
0.6420	<u>0.1928</u>
0.6515	0.1635
0.6596	<u>0.1602</u>
0.6672	0.0992
0.6751	<u>0.1282</u>
0.6843	0.1438
0.6954	<u>0.1416</u>
0.7086	0.1400
0.7240	<u>0.1412</u>
0.7412	0.1530
0.7595	<u>0.0795</u>
0.7780	0.1013
0.7956	<u>0.1150</u>
0.8115	0.1801
0.8247	<u>0.1082</u>
0.8345	0.0812
0.8407	<u>0.0347</u>
0.8431	0.0052
0.8423	<u>0.0017</u>
0.8387	0.0002
...	...
...	...

We can decompose the data into 64 pure sine waves using the **Discrete Fourier Transform (DFT)** - just the first few sine waves are shown.

The Fourier Coefficients are reproduced as a column of numbers (just the first 30 or so coefficients are shown).

Note that at this stage we have **not** done dimensionality reduction, we have merely changed the representation...

An Example of a Dimensionality Reduction Technique III



We have discarded
of the data.

$$\frac{15}{16}$$

Raw Data

- 0.4995
- 0.5264
- 0.5523
- 0.5761
- 0.5973
- 0.6153
- 0.6301
- 0.6420
- 0.6515
- 0.6596
- 0.6672
- 0.6751
- 0.6843
- 0.6954
- 0.7086
- 0.7240
- 0.7412
- 0.7595
- 0.7780
- 0.7956
- 0.8115
- 0.8247
- 0.8345
- 0.8407
- 0.8431
- 0.8423
- 0.8387
- ...
- ...

Fourier Coefficients

- 1.5698
- 1.0485
- 0.7160
- 0.8406
- 0.3709
- 0.4670
- 0.2667
- 0.1928
- 0.1635
- 0.1602
- 0.0992
- 0.1282
- 0.1438
- 0.1416
- 0.1400
- 0.1412
- 0.1530
- 0.0795
- 0.1013
- 0.1150
- 0.1801
- 0.1082
- 0.0812
- 0.0347
- 0.0052
- 0.0017
- 0.0002
- ...
- ...
- ...

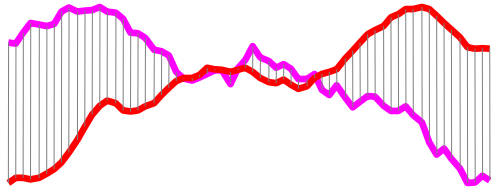
Truncated Fourier Coefficients

- 1.5698
- 1.0485
- 0.7160
- 0.8406
- 0.3709
- 0.4670
- 0.2667
- 0.1928

$n = 128$
 $N = 8$
 $C_{\text{ratio}} = 1/16$

... however, note that the first few sine waves tend to be the largest (equivalently, the magnitude of the Fourier coefficients tend to decrease as you move down the column). We can therefore truncate most of the small coefficients with little effect.

An Example of a Dimensionality Reduction Technique III



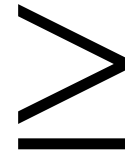
$$D(Q, C) \equiv \sqrt{\sum_{i=1}^n (q_i - c_i)^2}$$

Raw Data 1 **Raw Data 2**

0.4995	-	0.7412
0.5264	-	0.7595
0.5523	-	0.7780
0.5761	-	0.7956
0.5973	-	0.8115
0.6153	-	0.8247
0.6301	-	0.8345
0.6420	-	0.8407
0.6515	-	0.8431
0.6596	-	0.8423
0.6672	-	0.8387
0.6751	-	0.4995
0.6843	-	0.5264
0.6954	-	0.5523
0.7086	-	0.5761
0.7240	-	0.5973
0.7412	-	0.6153
0.7595	-	0.6301
0.7780	-	0.6420
0.7956	-	0.6515
0.8115	-	0.6596
0.8247	-	0.6672
0.8345	-	0.6751
0.8407	-	0.6843
0.8431	-	0.6954
0.8423	-	0.7086
0.8387	-	0.7240
...		...
...		...
...		...

Truncated Fourier Coefficients 1

Truncated Fourier Coefficients 2



<u>1.5698</u>	-	<u>1.1198</u>
<u>1.0485</u>	-	<u>1.4322</u>
<u>0.7160</u>	-	<u>1.0100</u>
<u>0.8406</u>	-	<u>0.4326</u>
<u>0.3709</u>	-	<u>0.5609</u>
<u>0.4670</u>	-	<u>0.8770</u>
<u>0.2667</u>	-	<u>0.1557</u>
<u>0.1928</u>	-	<u>0.4528</u>

The Euclidean distance between the two truncated Fourier coefficient vectors is always less than or equal to the Euclidean distance between the two raw data vectors*.

So **DFT** allows lower bounding!

*Parseval's Theorem

Mini Review for the Generic Data Mining Algorithm

We *cannot* fit all that raw data in main memory.

We *can* fit the dimensionally reduced data in main memory.

So we will solve the problem at hand on the dimensionally reduced data, making a few accesses to the raw data were necessary, and, if we are careful, the lower bounding property will insure that we get the right answer!

Raw Data 1	Raw Data 2	Raw Data n
4995	0.7412	0.8115
5264	0.7595	0.8247
5523	0.7780	0.8345
5761	0.7956	0.8407
5973	0.8115	0.8431
6153	0.8247	0.8423
6301	0.8345	0.8387
6420	0.8407	0.4995
6515	0.8431	0.7412
6596	0.8423	0.7595
6672	0.8387	0.7780
6751	0.4995	0.7956
6843	0.5264	0.5264
6954	0.5523	0.5523
7086	0.5761	0.5761
7240	0.5973	0.5973
7412	0.6153	0.6153

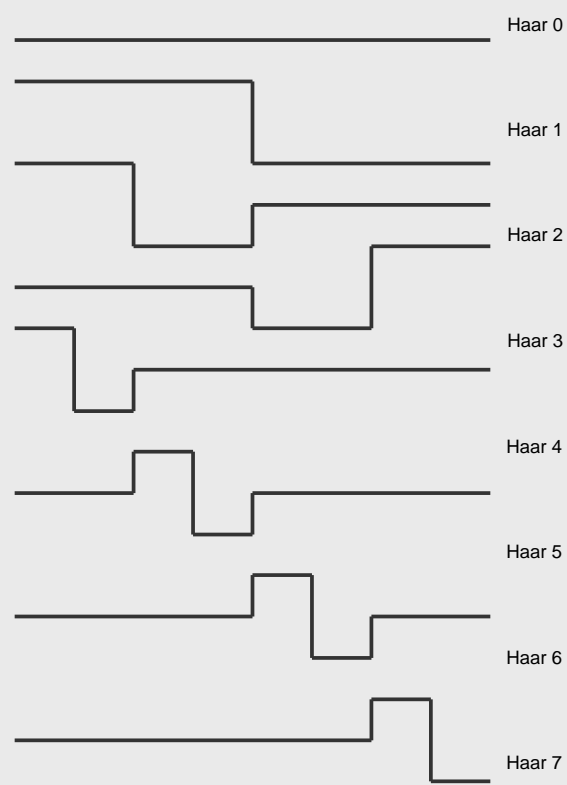
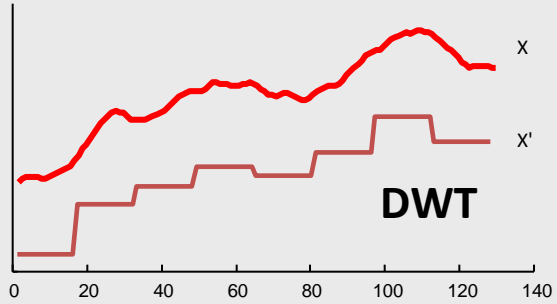
← Disk

Main
Memory



Truncated Fourier Coefficients 1	Truncated Fourier Coefficients 2	Truncated Fourier Coefficients n
1.5698	1.1198	1.3434
<u>1.0485</u>	<u>1.4322</u>	<u>1.4343</u>
0.7160	1.0100	1.4643
<u>0.8406</u>	<u>0.4326</u>	<u>0.7635</u>
0.3709	0.5609	0.5448
<u>0.4670</u>	<u>0.8770</u>	<u>0.4464</u>
0.2667	0.1557	0.7932
<u>0.1928</u>	<u>0.4528</u>	<u>0.2126</u>

Discrete Wavelet Transform I



Basic Idea: Represent the time series as a linear combination of Wavelet basis functions, but keep only the first N coefficients.

Although there are many different types of wavelets, researchers in time series mining/indexing generally use Haar wavelets.

Haar wavelets seem to be as powerful as the other wavelets for most problems and are very easy to code.

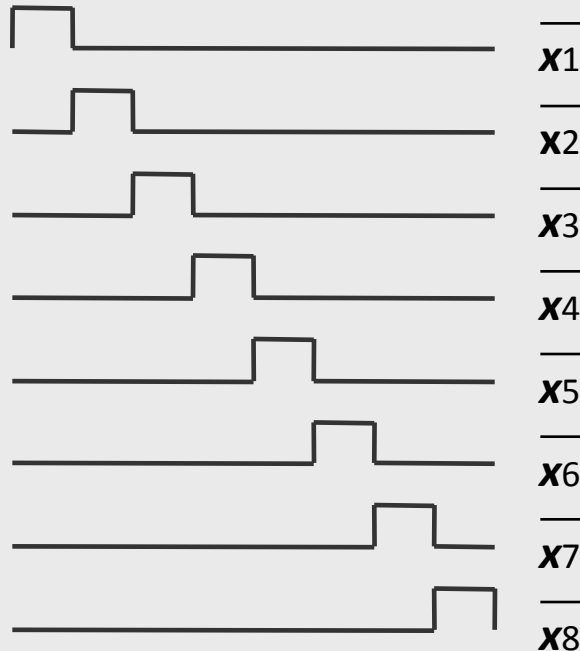
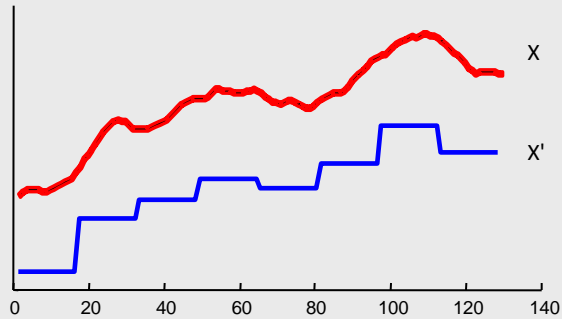
Excellent free Wavelets Primer

Stollnitz, E., DeRose, T., & Salesin, D. (1995). *Wavelets for computer graphics A primer: IEEE Computer Graphics and Applications*.



Alfred Haar
1885-1933

Piecewise Aggregate Approximation I



Basic Idea: Represent the time series as a sequence of N box basis functions.

Note that each box is of the same length (n/N).

$$\bar{x}_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} x_j$$

Given the reduced dimensionality representation we can calculate the approximate Euclidean distance as...

$$DR(\bar{X}, \bar{Y}) \equiv \sqrt{\frac{n}{N}} \sqrt{\sum_{i=1}^N (\bar{x}_i - \bar{y}_i)^2}$$

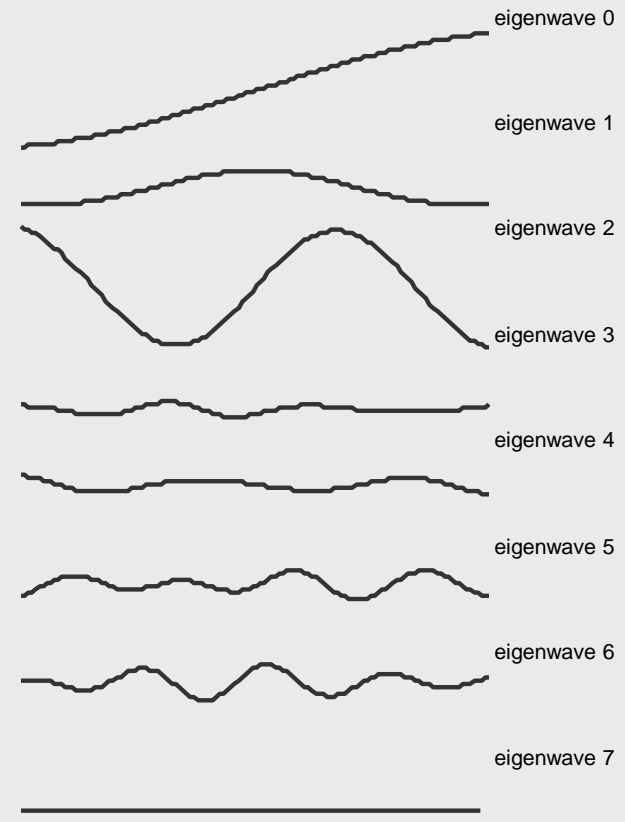
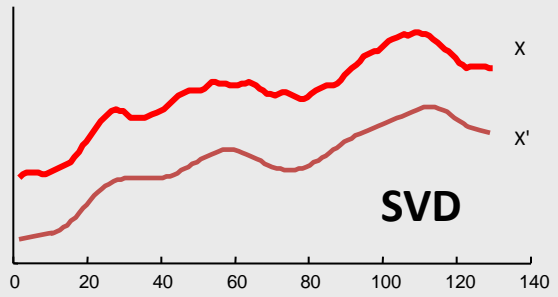
This measure is provably lower bounding.

Independently introduced by two authors

- Keogh, Chakrabarti, Pazzani & Mehrotra, KAIS (2000) / Keogh & Pazzani PAKDD April 2000

- Byoung-Kee Yi, Christos Faloutsos, VLDB September 2000

Singular Value Decomposition I



Basic Idea: Represent the time series as a linear combination of *eigenwaves* but keep only the first *N* coefficients.

SVD is similar to Fourier and Wavelet approaches is that we represent the data in terms of a linear combination of shapes (in this case *eigenwaves*).

SVD differs in that the *eigenwaves* are data dependent.

SVD has been successfully used in the text processing community (where it is known as *Latent Symantec Indexing*) for many years.

[Good free SVD Primer](#)

Singular Value Decomposition - A Primer.
Sonia Leach



James Joseph Sylvester
1814-1897

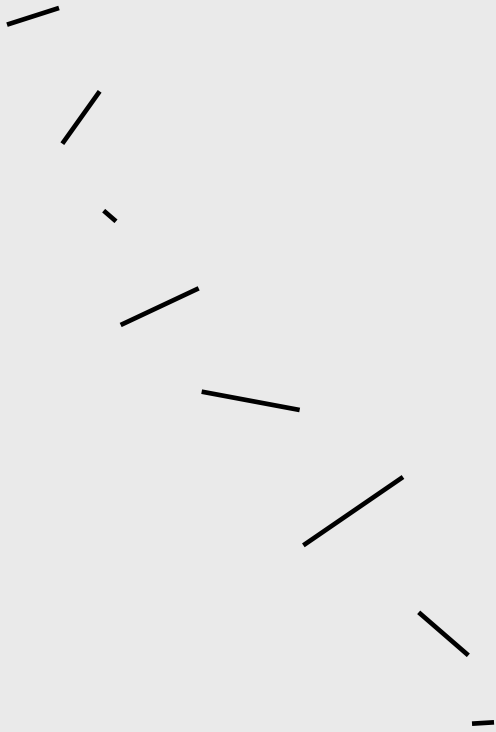
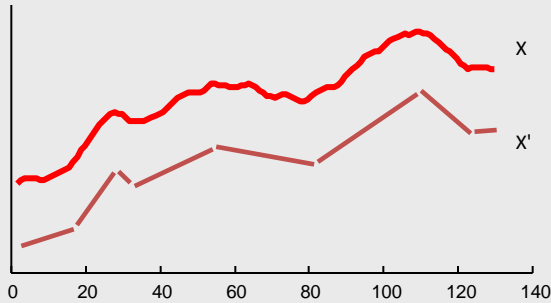


Camille Jordan
(1838--1921)



Eugenio Beltrami
1835-1899

Piecewise Linear Approximation



Basic Idea: Represent the time series as a sequence of straight lines.

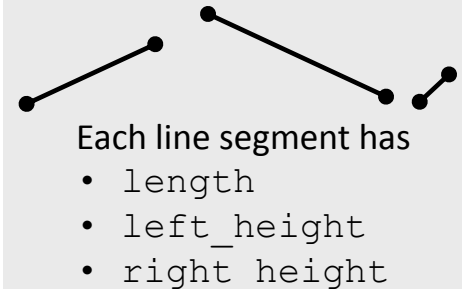
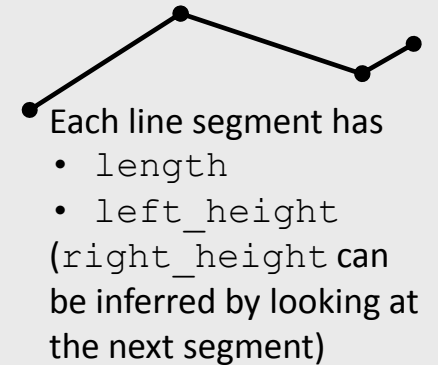
Lines could be **connected**, in which case we are allowed $N/2$ lines

If lines are **disconnected**, we are allowed only $N/3$ lines

Personal experience on dozens of datasets suggest **disconnected** is better. Also only **disconnected** allows a lower bounding Euclidean approximation

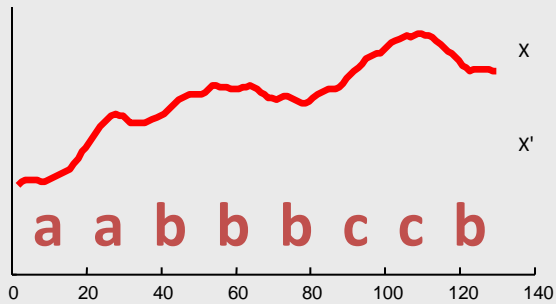


Karl Friedrich Gauss, 1777 - 1855



- Good ability to compress natural signals.
- Fast linear time algorithms for PLA exist.
- Already widely accepted in some communities (e.g. biomedical)

Symbolic Approximation I



a

a

b

b

b

c

c

b

0

1

2

3

4

5

6

7

Basic Idea: Convert the time series into an alphabet of discrete symbols. Use string indexing techniques to manage the data.

Potentially an interesting idea, but all work thus far are very ad hoc.

Pros and Cons of Symbolic Approximation as a time series representation.

- Potentially, we could take advantage of a wealth of techniques from the very mature field of string processing and bioinformatics.
- It is not clear how we should discretize the times series (discretize the values, the slope, shapes? How big of an alphabet? etc).
- There are more than 210 different variants of this, at least 35 in data mining conferences.

Summary of Time Series Similarity

- If you have **short time series**, use DTW after searching over the warping window size¹ (and shape²)
- Then use envelope based lower bounds to speed things up³.
- If you have **long time series**, and you know nothing about your data, **try compression based dissimilarity**.
- If you do **know something** about your data, try to leverage of this knowledge to **extract features**.