



Similarity Search on Time Series Data: Past, Present and Future

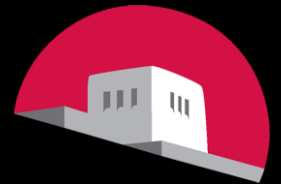
Abdullah Mueen

Assistant Professor of Computer Science
University of New Mexico
mueen@unm.edu



Download slides while waiting:

<http://www.cs.unm.edu/~mueen/Tutorial/CIKM2016Tutorial.pdf>



Acknowledgements

- The following people helped with advice slidea or images: Amanda Minnich, Sindhuja Sivakumar, Diego Silva, François Petitjean, Gustavo Batista, Michail Vlachos, George Kollios, Dimitrios Gunopulos, Eamonn Keogh (If I forgot anyone, apologies)
- I also acknowledge NSF CCF # 1527127. Any error or opinion is solely my responsibility
- My GOAL: Present a survey of literature on similarity search on time series data
- There will be a Q/A segment after every hour

What are Time Series? 1 of 2

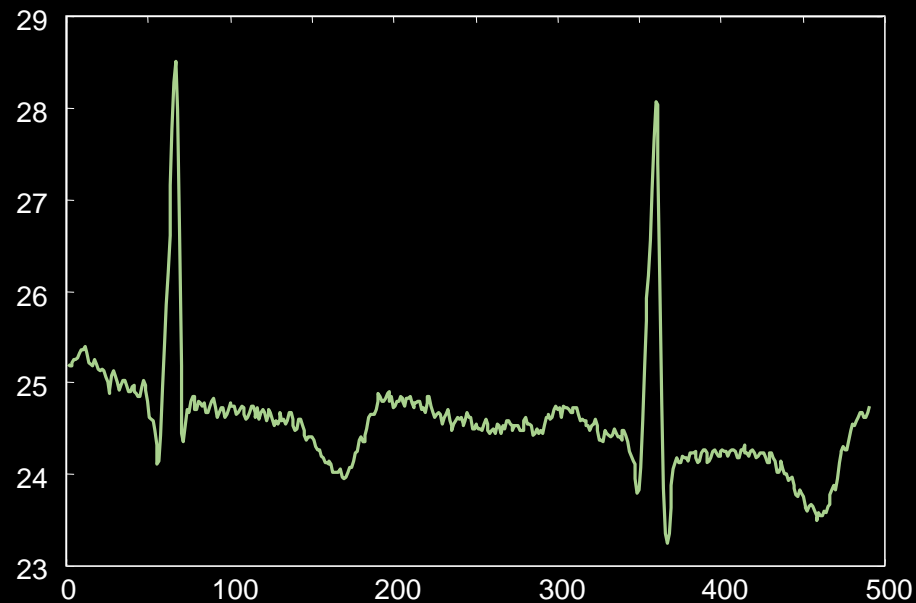
A time series is a collection of observations made sequentially in time.

More than most types of data, time series lend themselves to *visual* inspection and intuitions...

For example, looking at the numbers in this green vector tells us nothing.
But after *plotting* the data, we can recognize a heartbeat, and possibly even diagnose this person's disease.

This tutorial will leverage the visual intuitiveness of time series.

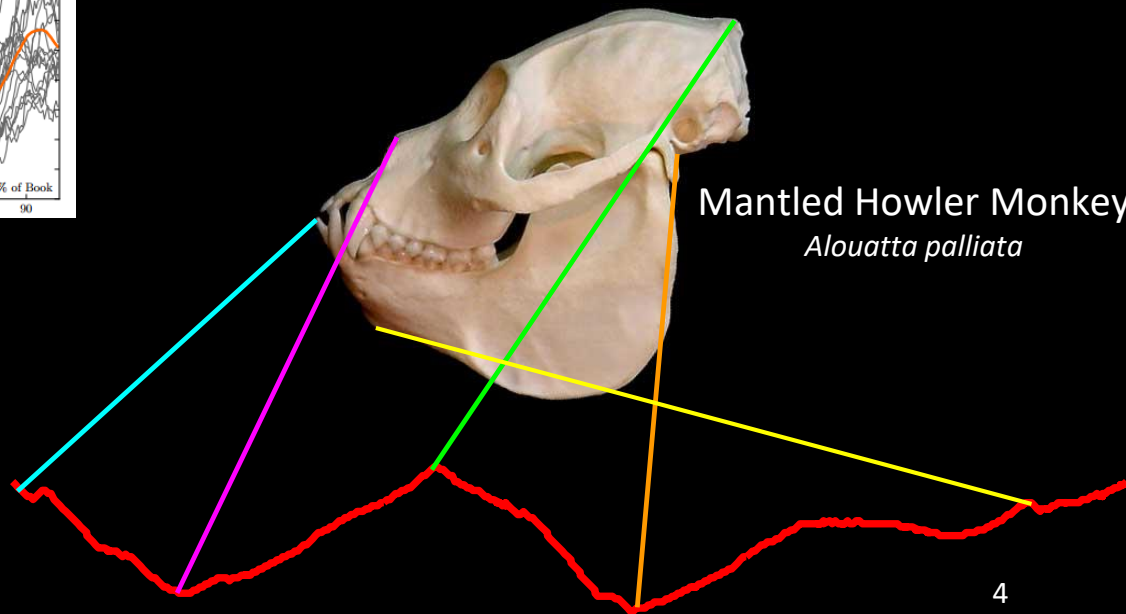
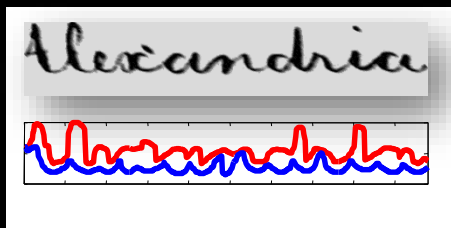
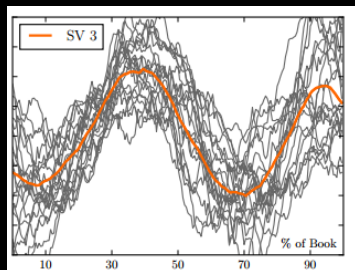
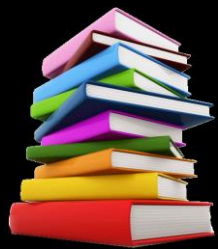
25.350
25.350
25.400
25.400
25.325
25.225
25.200
25.175
..
24.625
24.675
24.675
24.675



What are Time Series? 2 of 2

As an aside... (not the main point for today)

Many types of data that are not *true* time series can be fruitfully transformed into time series, including DNA, speech, textures, core samples, ASCII text, historical handwriting, novels and even *shapes*.



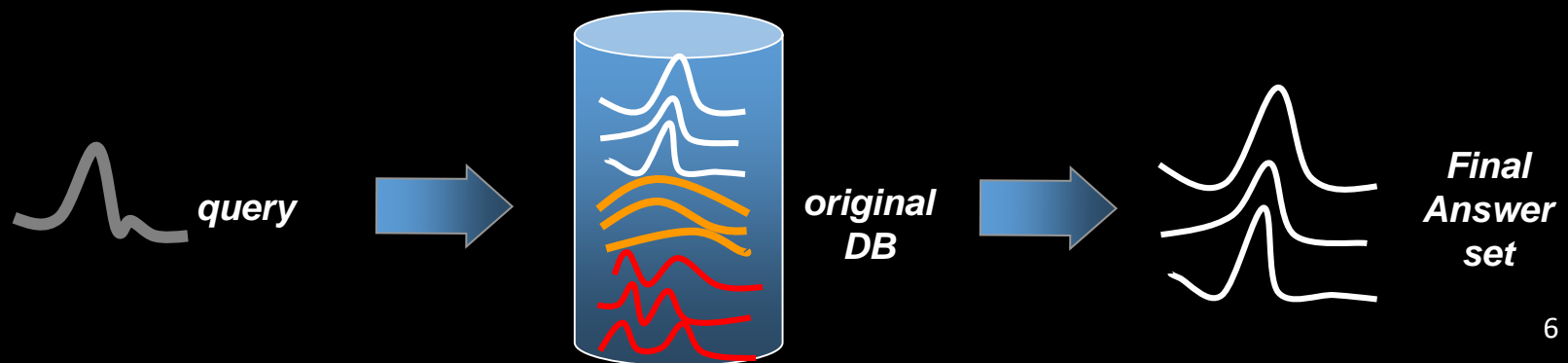
Mantled Howler Monkey
Alouatta palliata

Similarity Measures

- A similarity measure compares two time series and produces a number representing their similarity
 - A distance measure is an inverted similarity measure
- **Lockstep Measures**
 - Euclidean Distance
 - Correlation Coefficient
 - Cross-correlation
- **Elastic Measures**
 - Dynamic Time Warping
 - Edit Distance
 - Longest Common Subsequence

Similarity Search

- A query Q is given of length m
- n independent candidate time series C_1, C_2, \dots, C_n
- Nearest Neighbor query
 - Find K nearest neighbor of Q under a distance measure d
- Range Query
 - Find all time series C where $d(Q, C) \leq \epsilon$
- Density Estimation
 - Count the number of time series C where $d(Q, C) \leq \epsilon$



Why Similarity Search?

- Most data mining algorithms (e.g. clustering, classification, outlier detection, etc.) use similarity measures and similarity search as **subroutines**
- **Interpretable** mining of heterogeneous data objects is possible through similarity search
- Great way to **embed scientific knowledge** in mining algorithms by modifying and adapting similarity measures

Top-level Outline

- Similarity Measures
- Similarity Search
 - Lock-step search (e.g. Correlation Search)
 - Elastic search (e.g. DTW search)

Euclidean Distance Metric

Given two time series

$$\mathbf{x} = x_1 \dots x_n$$

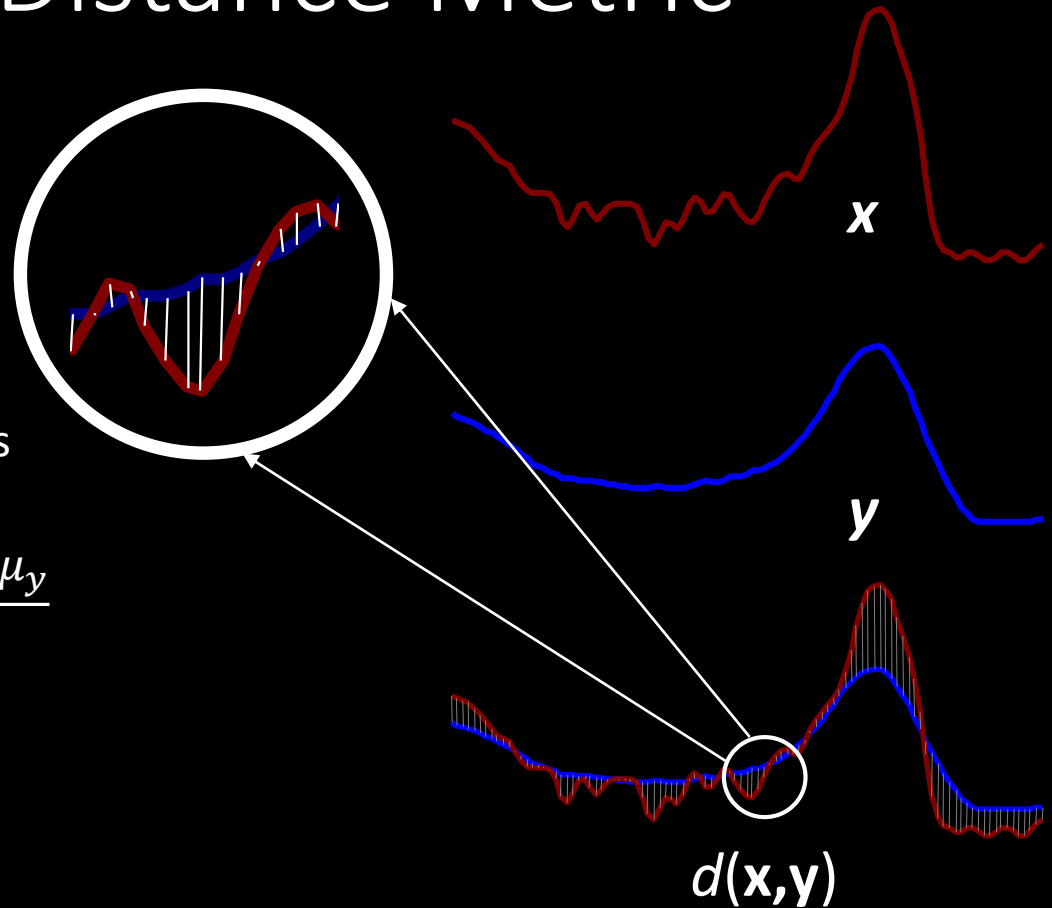
and

$$\mathbf{y} = y_1 \dots y_n$$

their z-Normalized Euclidean distance is defined as:

$$\hat{x}_i = \frac{x_i - \mu_x}{\sigma_x} \quad \hat{y}_i = \frac{y_i - \mu_y}{\sigma_y}$$

$$d(x, y) = \sqrt{\sum_{i=1}^n (\hat{x}_i - \hat{y}_i)^2}$$



Pearson's Correlation Coefficient

- Given two time series x and y of length m .
- Correlation Coefficient:

$$\text{corr}(x, y) = \frac{\sum_{i=1}^m x_i y_i - m\mu_x \mu_y}{m\sigma_x \sigma_y}$$

- Where $\mu_x = \frac{\sum_{i=1}^m x_i}{m}$ and $\sigma_x^2 = \frac{\sum_{i=1}^m x_i^2}{m} - \mu_x^2$

- Sufficient Statistics:

$$\sum_{i=1}^m x_i y_i \quad \sum_{i=1}^m x_i \quad \sum_{i=1}^m y_i \quad \sum_{i=1}^m x_i^2 \quad \sum_{i=1}^m y_i^2$$

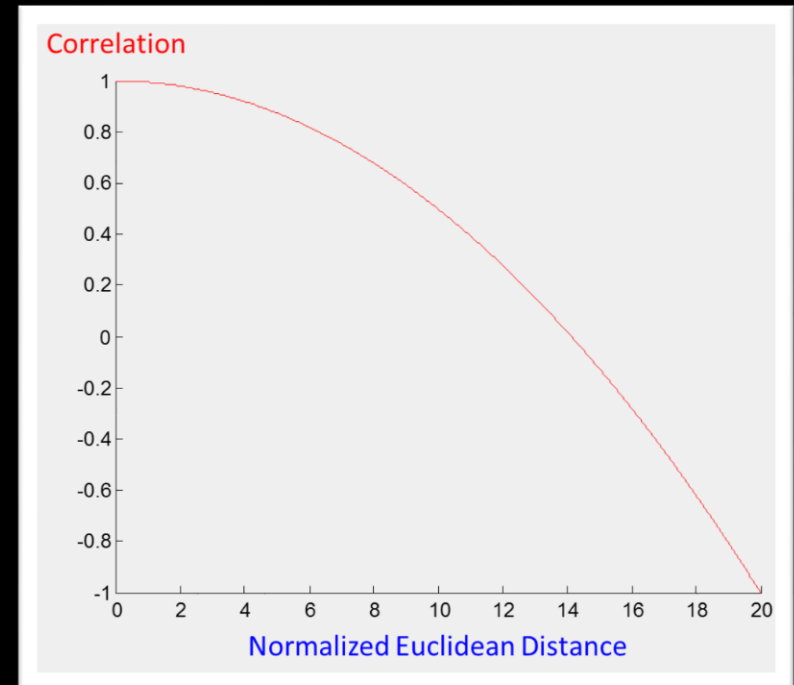
One linear scan

Relationship with Euclidean Distance

$$d(\hat{\mathbf{x}}, \hat{\mathbf{y}}) = \sqrt{2n(1 - \text{corr}(\mathbf{x}, \mathbf{y}))}$$

$$\hat{x}_i = \frac{x_i - \mu_x}{\sigma_x} \text{ and } \hat{y}_i = \frac{y_i - \mu_y}{\sigma_y}$$

$$d^2(\hat{\mathbf{x}}, \hat{\mathbf{y}}) = \sum_{i=1}^n (\hat{x}_i - \hat{y}_i)^2$$

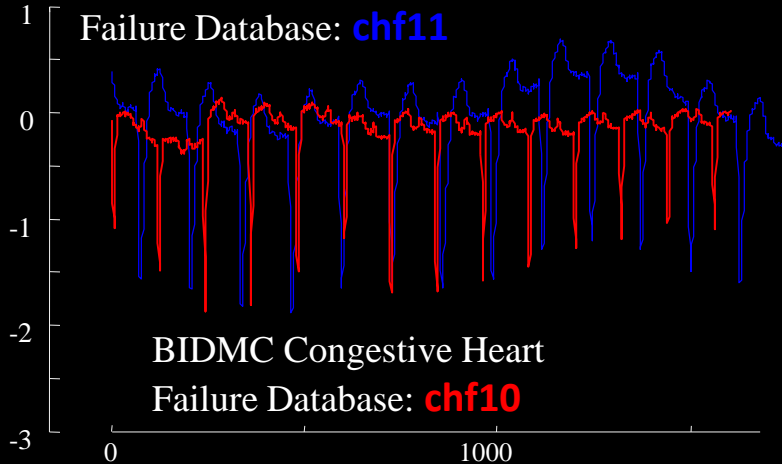


- Correlation coefficient does not obey triangular inequality
- Maximizing correlation coefficient can be achieved by minimizing normalized Euclidean distance and vice versa

The Importance of z-Normalization and correlation 1 of 2

BIDMC Congestive Heart

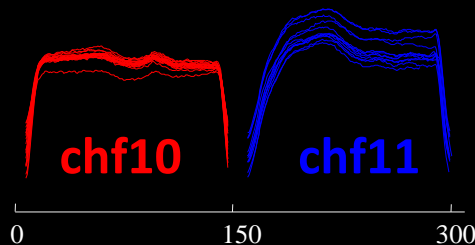
Failure Database: **chf11**



BIDMC Congestive Heart

Failure Database: **chf10**

Extracted
beats →



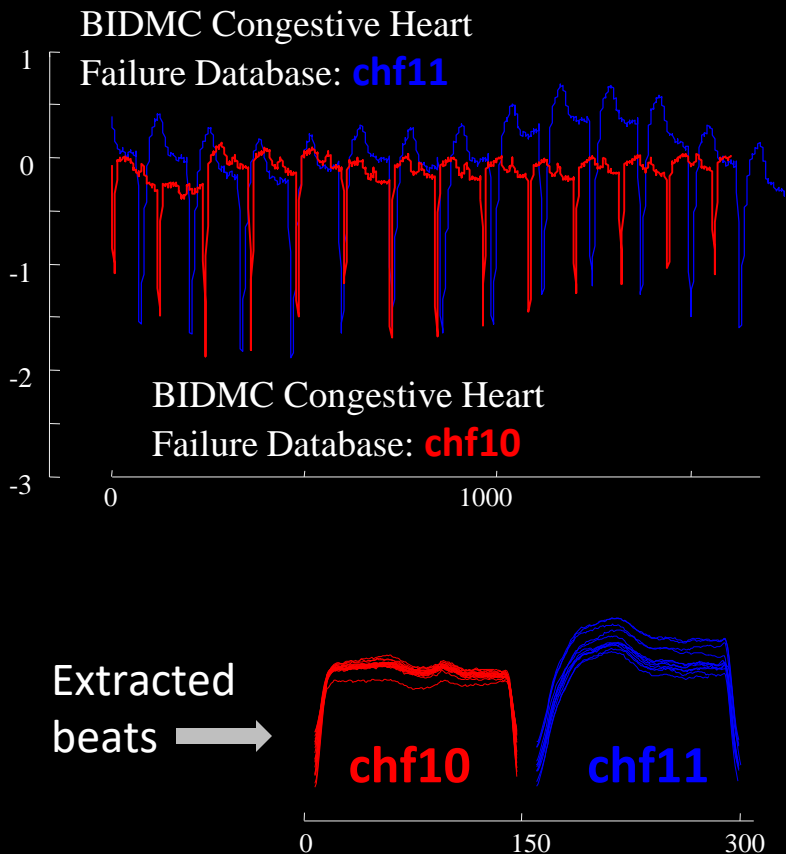
Essentially all datasets must have *every* subsequence z-normalized.

There are a handful of occasions where it does not make sense to z-normalize, but in those cases, similarity search does not make sense either.

In this example, we begin by extracting heartbeats from two unrelated people.

Even without normalization, it happens that both sets have almost the same mean and standard deviation. Given that, do we need to bother to normalize them? (next slide)

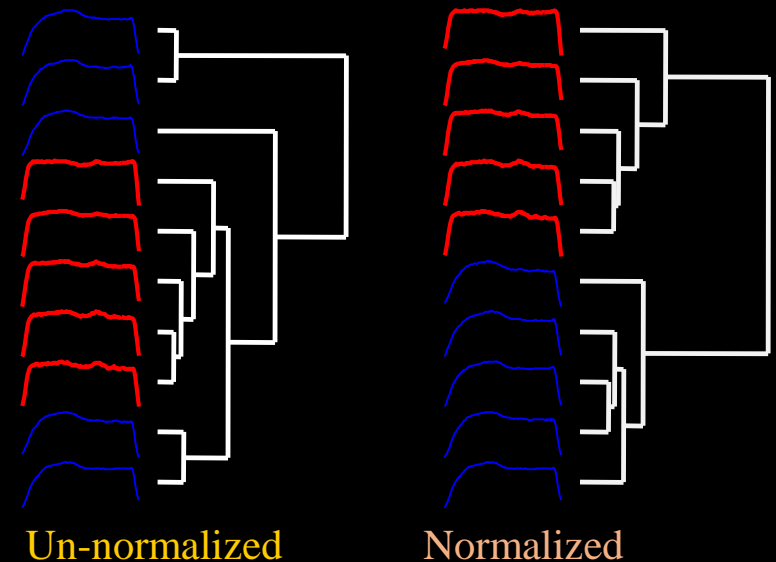
The Importance of z-Normalization and correlation 2 of 2



Surprisingly z-normalizing can be a computational bottleneck, but later we will show you how to fix that.

Without normalization, the results are very poor, some blue heartbeats are closer to red heartbeats than there are to another blue beat.

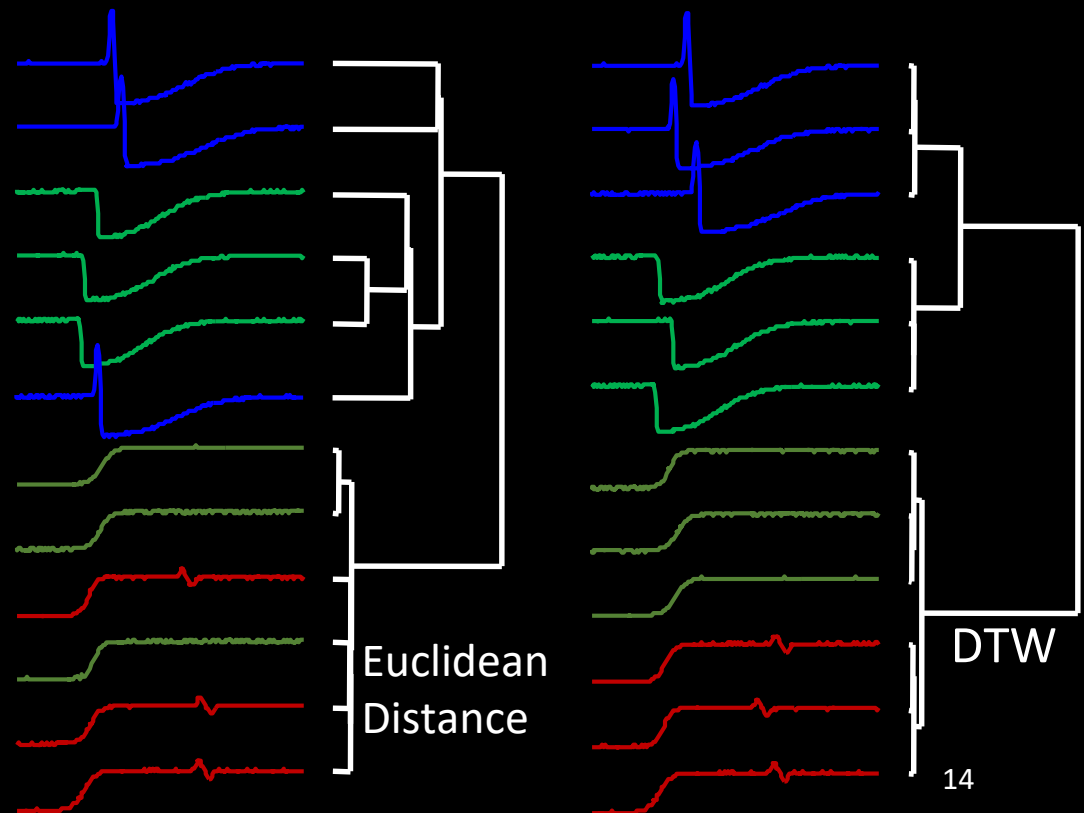
With normalization, the results are perfect.



In this example, we extracted heartbeats from two different time series, and clustered them with and without normalization.

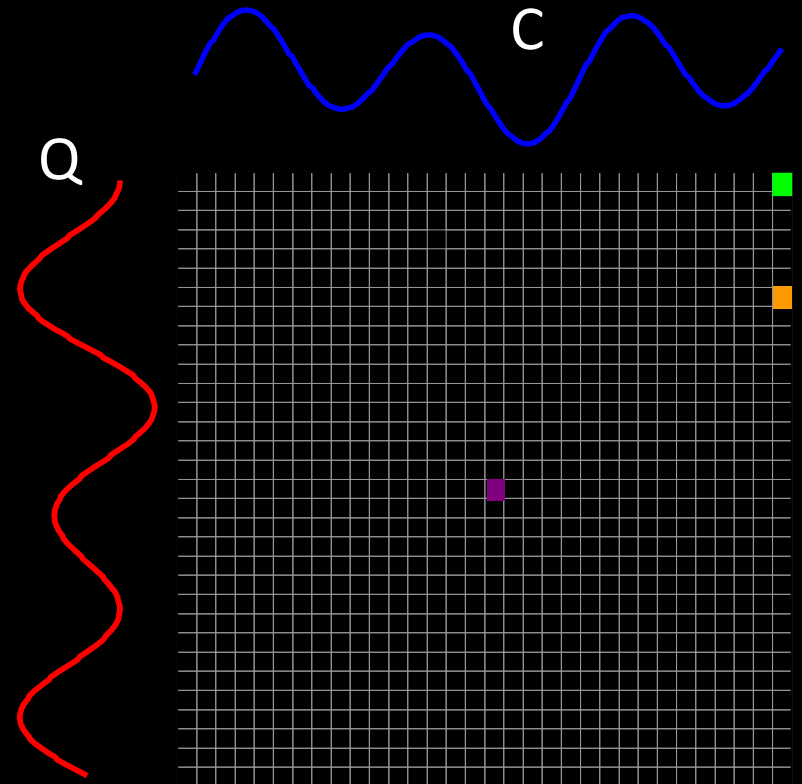
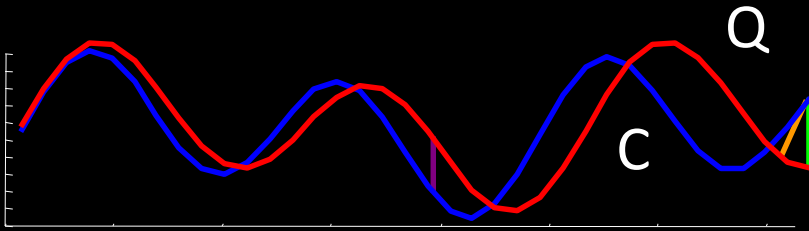
What is Dynamic Time Warping?

- DTW is an algorithm for measuring similarity between two time series which may vary (i.e. *warp*) in timing.
- This invariance to warping is critical in many domains, for many tasks.
- Without warping invariance, we are often condemned to very poor results.



How is DTW Calculated? I

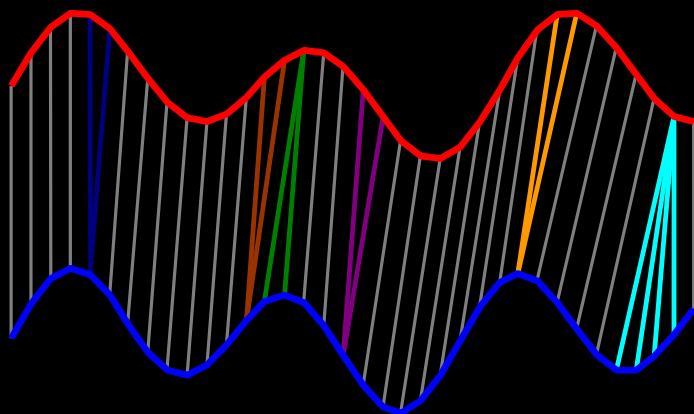
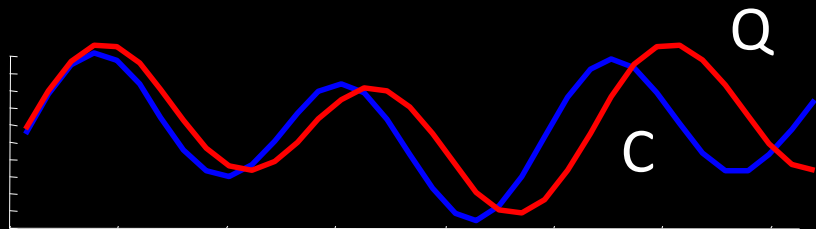
We create a matrix the size of $|Q|$ by $|C|$, then fill it in with the distance between every possible pair of points in our two time series.



How is DTW Calculated? II

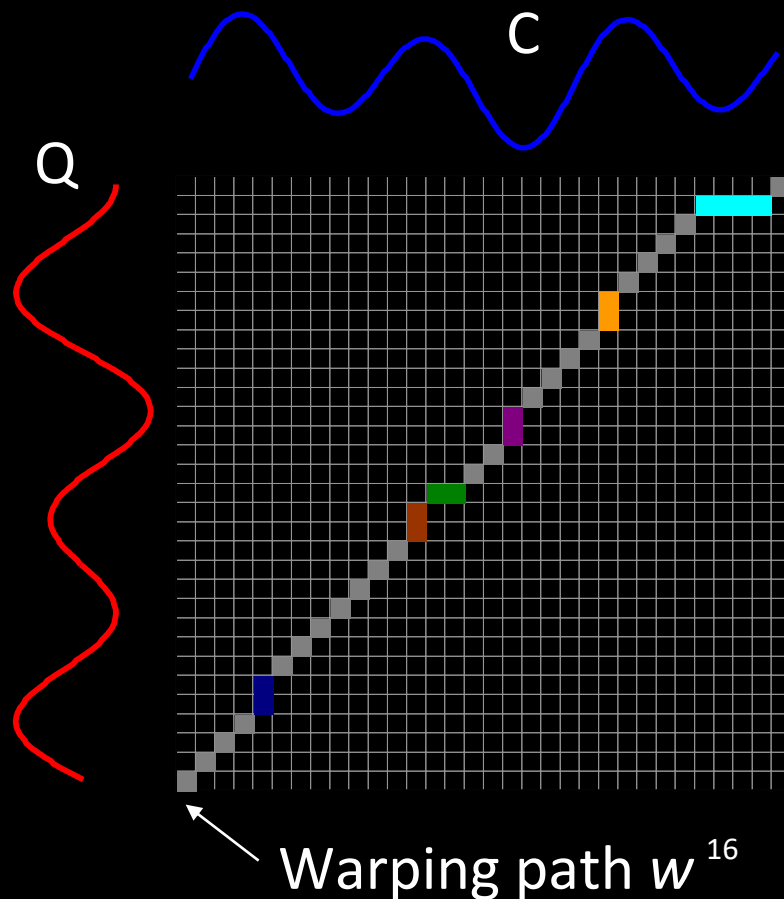
Every possible warping between two time series, is a path through the matrix.
We want the *best* one...

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

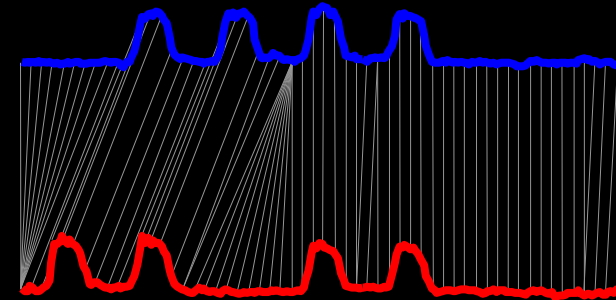
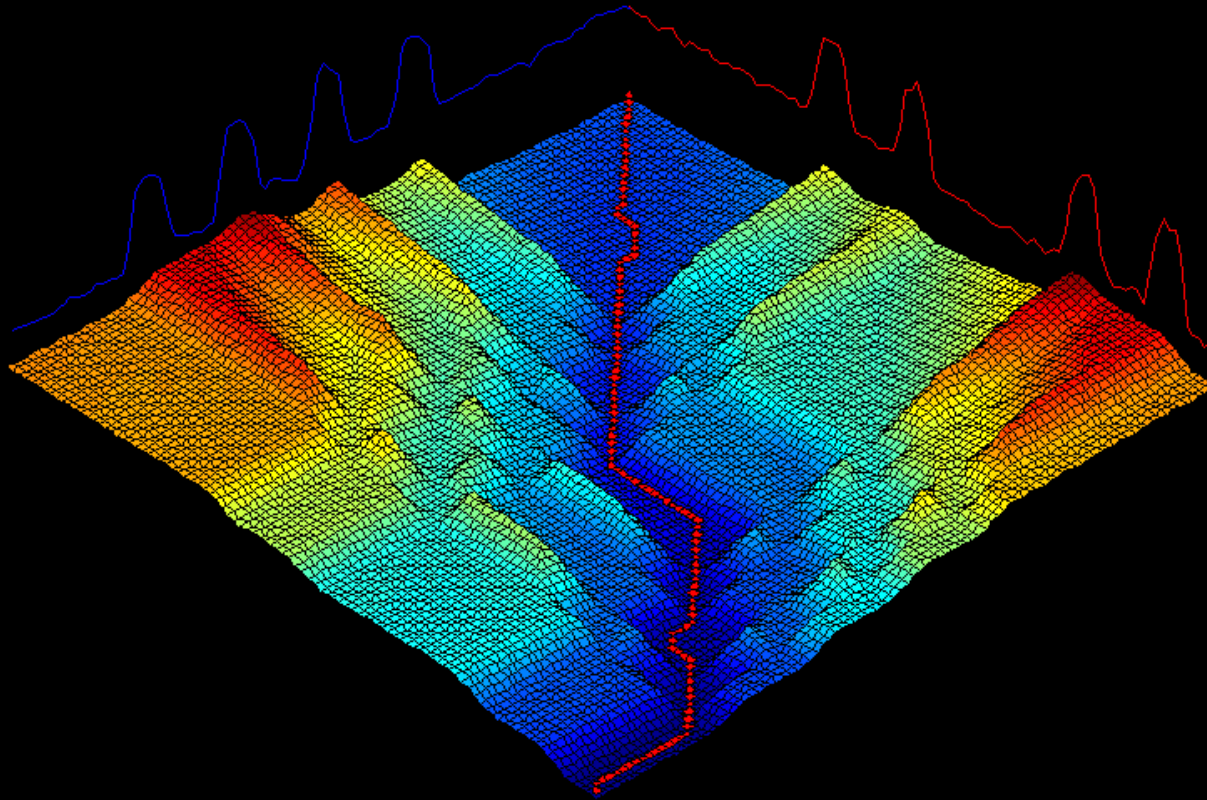


This recursive function gives us the minimum cost path

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



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



This example shows two one-week periods from an electrical 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.

DTW is a Distance *Measure*, not a *Metric* 1 of 2

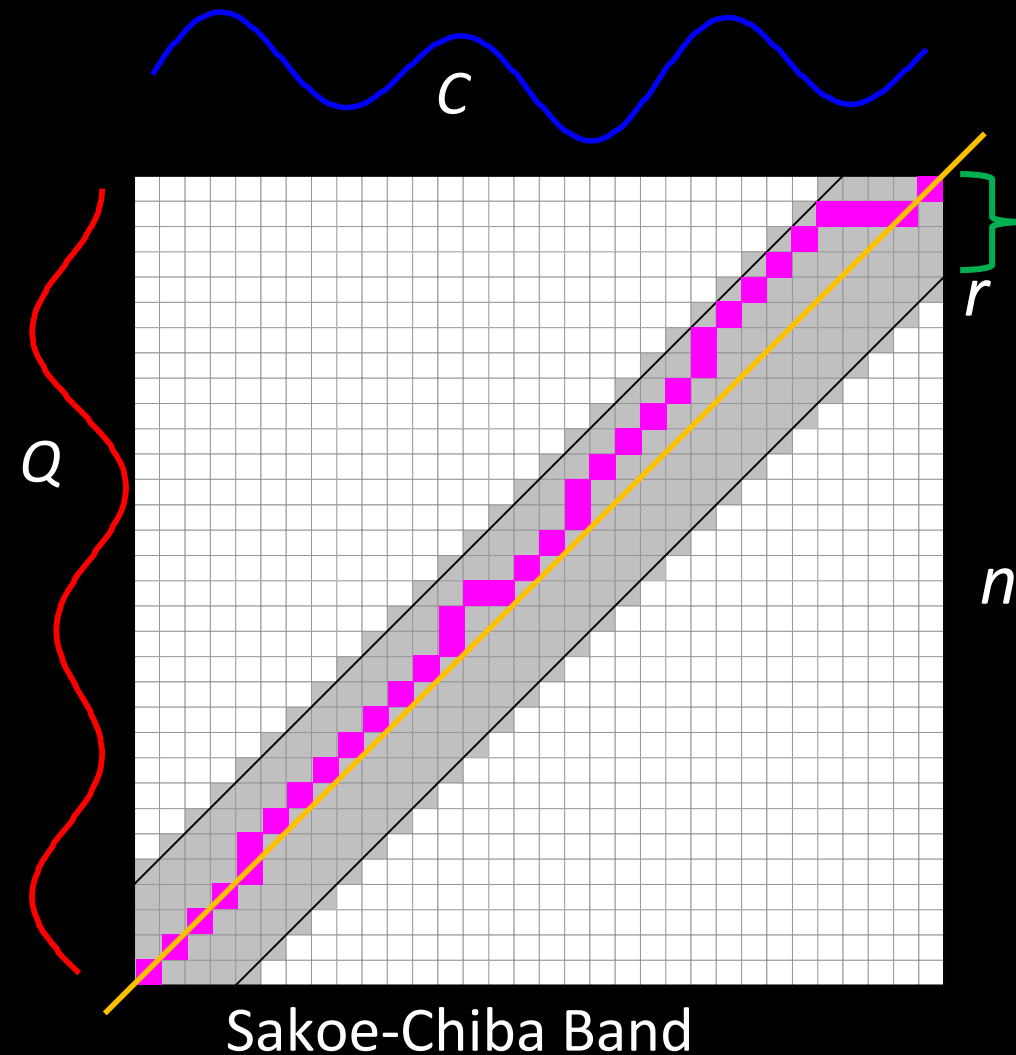
Requirements to be a metric

Yes for DTW	{	$D(A,B) = D(B,A)$	<i>Symmetry</i>
		$D(A,A) = 0$	<i>Constancy of Self-Similarity</i>
No for DTW	{	$D(A,B) = 0 \text{ iff } A=B$	<i>Positivity (Separation)</i>
		$D(A,B) \leq D(A,C) + D(B,C)$	<i>Triangular Inequality</i>

Normally we prefer metrics over measures for two reasons:

- Non-Metrics can sometimes give pathological solutions when clustering or classifying data etc.
- Almost all speed-up “tricks” for high dimensional data exploit the Triangular Inequality.

Understanding w , the Warping Constraint



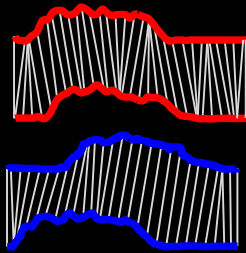
The value of w is the maximum amount the warping path is allowed to **deviate** from the **diagonal**.

It is normally expressed as the ratio $w = r/n$ (or as a percentage)

We need to understand w because:

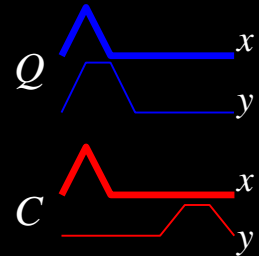
- The most useful speedup tricks all exploit w .
- The value chosen for w can greatly affect accuracy.

Generalizing to Multi-Dimensional Data 1 of 3



It is increasingly common to encounter Multi-Dimensional (MD) time series data. Here we measure the X-axis acceleration of both the left and right hand.

Given these pair of 2D objects...



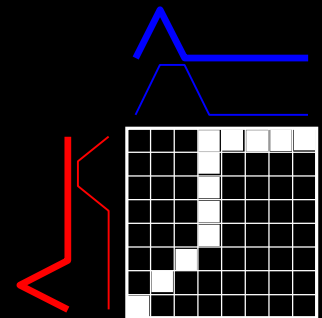
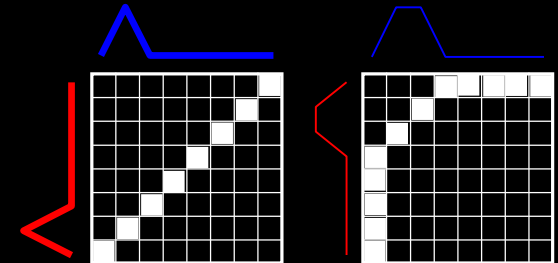
There are two obvious ways to compute the MD DTW score.

Independent: Just compute the DTW score for each dimension independently, and sum up each score.

$$DTW_I(Q, C) = DTW(Q_x, C_x) + DTW(Q_y, C_y) = 2.4$$

Dependent: Create a single distance matrix that reflect the distance between each corresponding pair of time series, then find the *single* warping path and distance as per normal.

$$DTW_D(Q, C) = DTW(\{Q_x, Q_y\}, \{C_x, C_y\}) = 3.2$$



Generalizing to Multi-Dimensional Data 2 of 3

So, of DTW_I and DTW_D which is best?

Lets think of it this way:

The thing we want classify is an physical process, the utterance of the word “*bicycle*”, the beat of a heart, an autograph, a tennis shot etc.

We cannot see the actual event, just 2 or more time series it created.

- If the physical process affects the time series simultaneously, then DTW_D will probably be best. We call this the *tightly coupled* case.
- If the physical process affects the time series with varying lags, then DTW_I will probably be best. We call this the *loosely coupled* case.

Example: Suppose we measure the directionless acceleration of the left and right wrists of a tennis player.

The “physical process” is a *backhand stroke*. If a two-handed backstroke, the two time series are tightly coupled. If a one-handed backhand, the two hands will be very loosely coupled .

Jo-Wilfried Tsonga



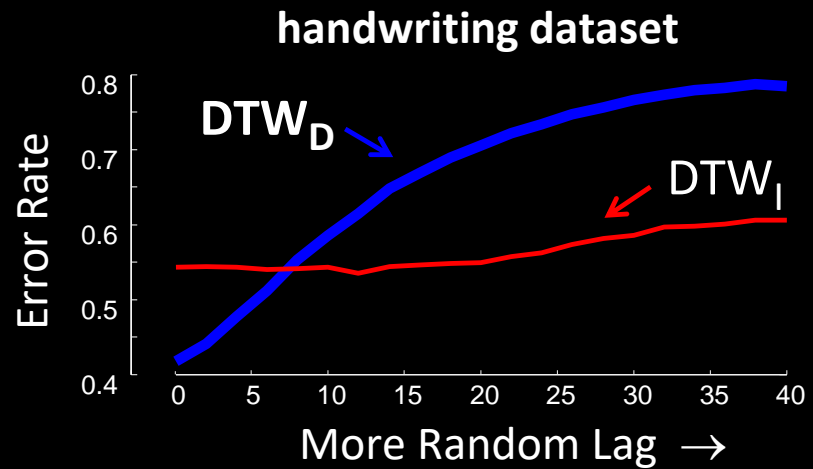
Generalizing to Multi-Dimensional Data 3 of 3

We can demonstrate the claim in the last slide with experiments.

Let us begin with a dataset that we are 100% sure is tightly coupled, then slowly add some random time lags into the data...

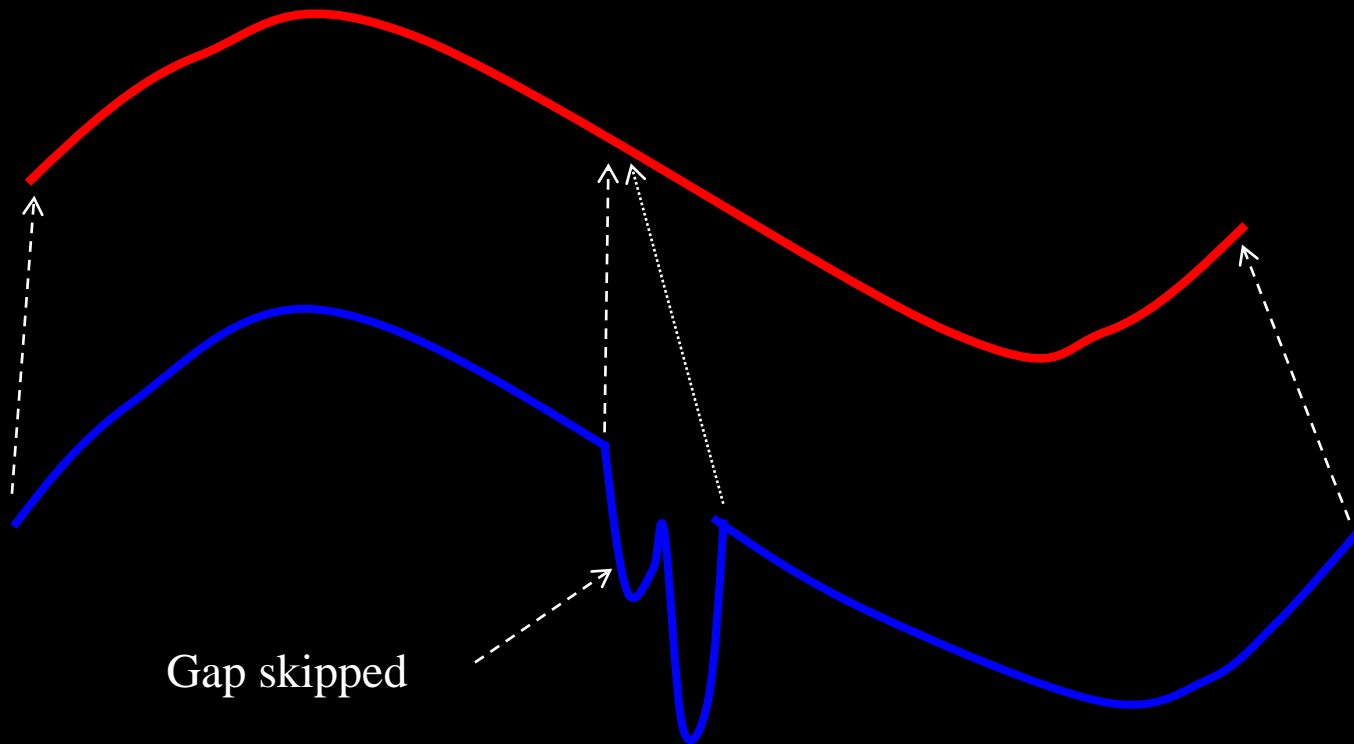
We know that the **handwriting dataset** is *tightly coupled*. We find that DTW_D has an error-rate of about 0.42, much better than that DTW_I has an error-rate of about 0.55.

However, as we uncouple the perfect synchronization by adding some random lag, DTW_D quickly gets worse, while that DTW_I is barely effected.



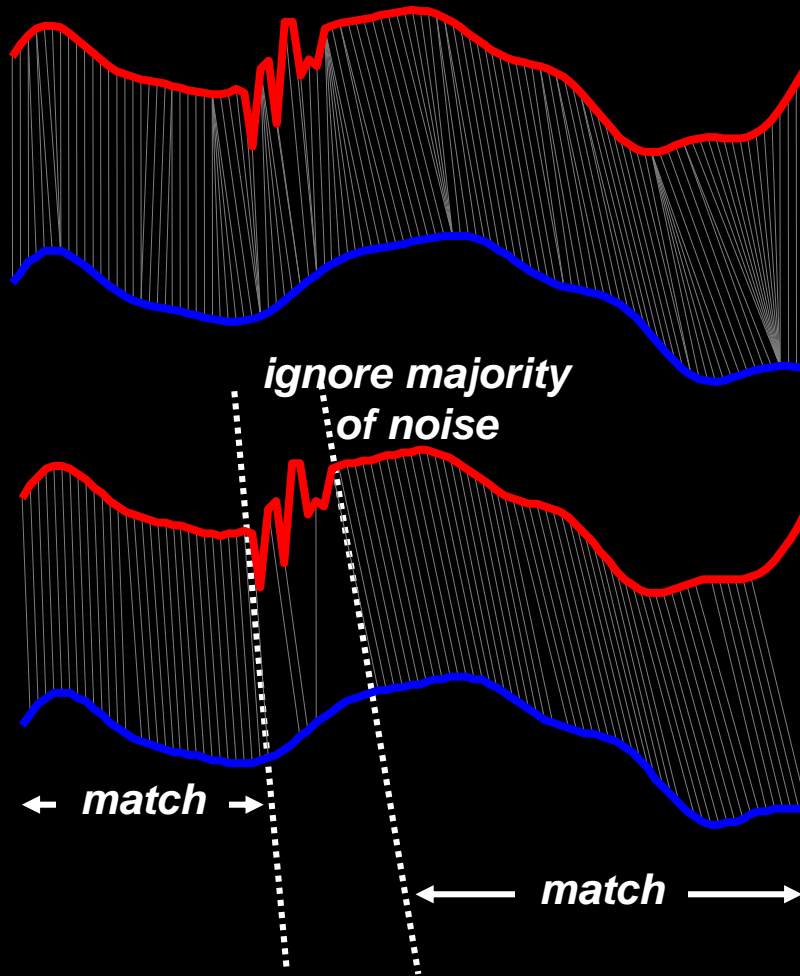
Longest Common Subsequence Measures

(Allowing for Gaps in Sequences)



Longest Common Subsequence (LCSS)

LCSS is more resilient to noise than DTW.



Disadvantages of DTW:

- A. All points are matched*
- B. Outliers can distort distance*
- C. One-to-many mapping*

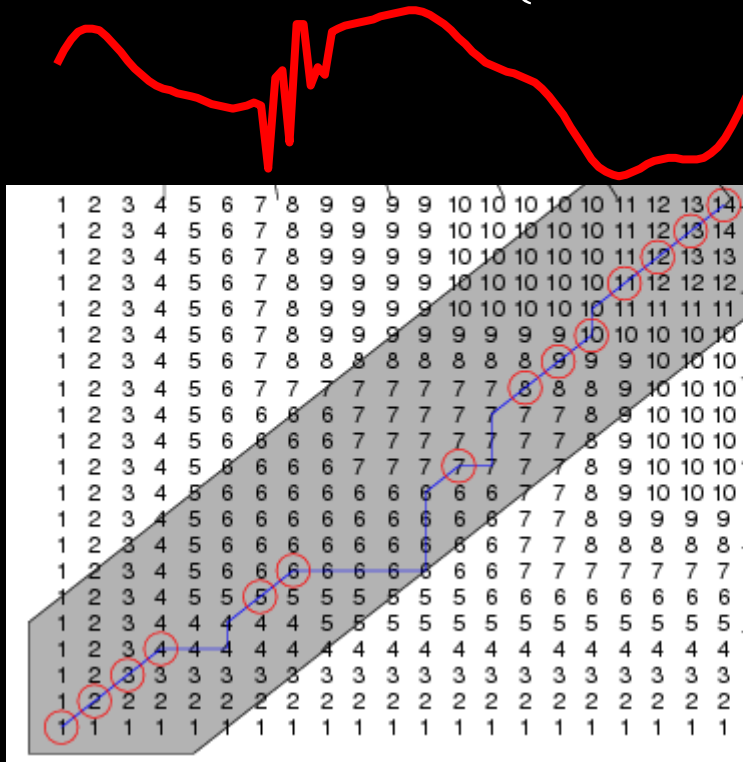
Advantages of LCSS:

- A. Outlying values not matched*
- B. Distance/Similarity distorted less*
- C. Constraints in time & space*

Longest Common Subsequence

*Similar dynamic programming solution as DTW, but now we measure **similarity not distance**.*

$$LCSS[i,j] = \begin{cases} 0 & \text{if } i = 0 \\ 0 & \text{if } j = 0 \\ 1 + LCSS[i-1, j-1] & \text{if } |a_{i,k} - b_{j,k}| < \epsilon \\ \max(LCSS[i-1, j], LCSS[i, j-1]) & \text{otherwise} \end{cases}$$



Can also be expressed as distance

$$D_{LCSS}(A, B) = 1 - \frac{LCSS_{\delta, \epsilon}(A, B)}{\min(n, m) \text{ or } \max(n, m)}$$

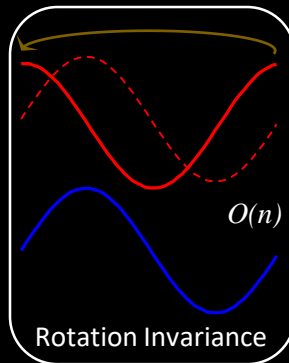
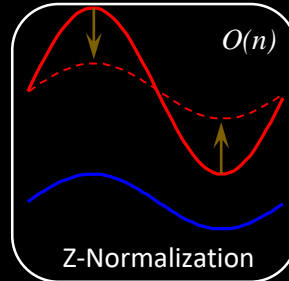
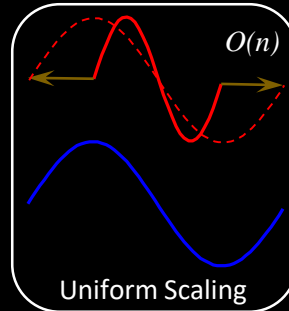
Various extensions of lock-step measures

- Preprocessing
- Alignment
- Postprocessing

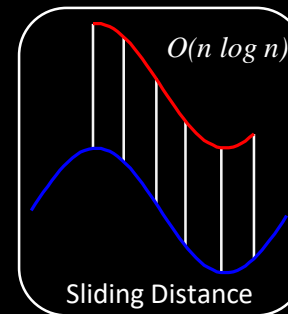
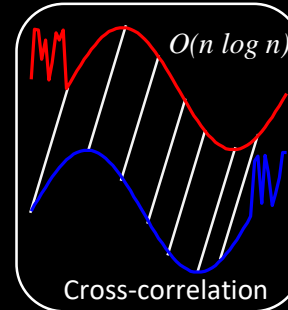
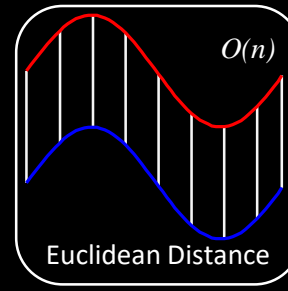
Let us assume that a sin wave should match with another sin wave without considering phase, amplitude, frequency, noise, etc.

1. Michail Vlachos, D. Gunopulos, and Gautam Das. 2004. Rotation invariant distance measures for trajectories. KDD '04. 707-712.
2. Dragomir Yankov, Eamonn J. Keogh, Jose Medina, Bill Yuan-chi Chiu, Victor B. Zordan: Detecting time series motifs under uniform scaling. KDD 2007: 844-853
3. Lexiang Ye, Eamonn J. Keogh: Time series shapelets: a new primitive for data mining. KDD 2009: 947-956
4. Gustavo E. A. P. A. Batista, Xiaoyue Wang, Eamonn J. Keogh: A Complexity-Invariant Distance Measure for Time Series. SDM 2011: 699-710

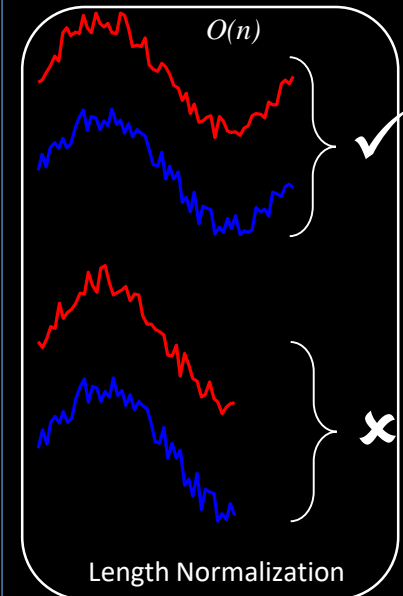
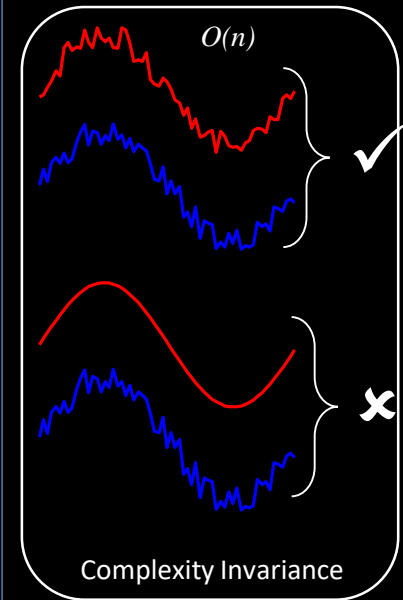
Preprocessing



Alignment



Postprocessing



Other Similarity Measures

- Move-Split-Merge (MSM) to achieve both dynamic alignment and **metric properties**
- MUNICH, PROUD, DUST: Similarity measure for **uncertain time series**

We will consider Euclidean distance and Dynamic Time Warping distance as representatives

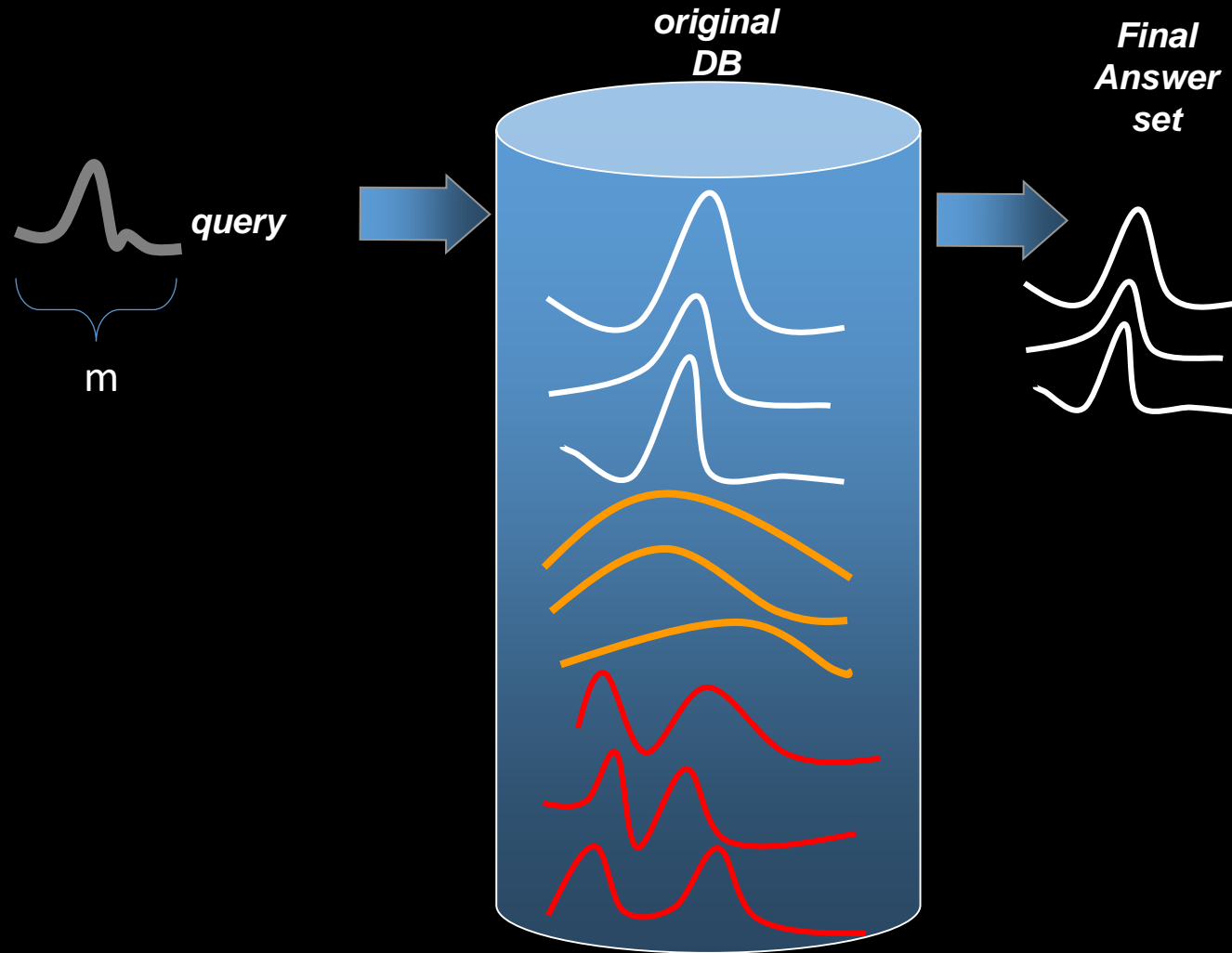
1. Alexandra Stefan, Vassilis Athitsos, Gautam Das: The Move-Split-Merge Metric for Time Series. IEEE Trans. Knowl. Data Eng. 25(6): 1425-1438 (2013)
2. Smruti R. Sarangi and Karin Murthy. 2010. DUST: a generalized notion of similarity between uncertain time series. KDD '10 , 383-392.
3. M. Yeh, K. Wu, P. Yu, and M. Chen. PROUD: a probabilistic approach to processing similarity queries over uncertain data streams. In EDBT, pages 684–695. ACM, 2009.
4. J. Aßfalg, H.-P. Kriegel, P. Kröger, and M. Renz. Probabilistic similarity search for uncertain time series. In SSDBM, pages 435–443, 2009.



Top-level Outline

- Similarity Measures
- Similarity Search
 - Lock-step search (e.g. Correlation Search)
 - Elastic search (e.g. DTW search)

Generic Search



Computational cost: $O(m \ n)$

GEMINI Framework*

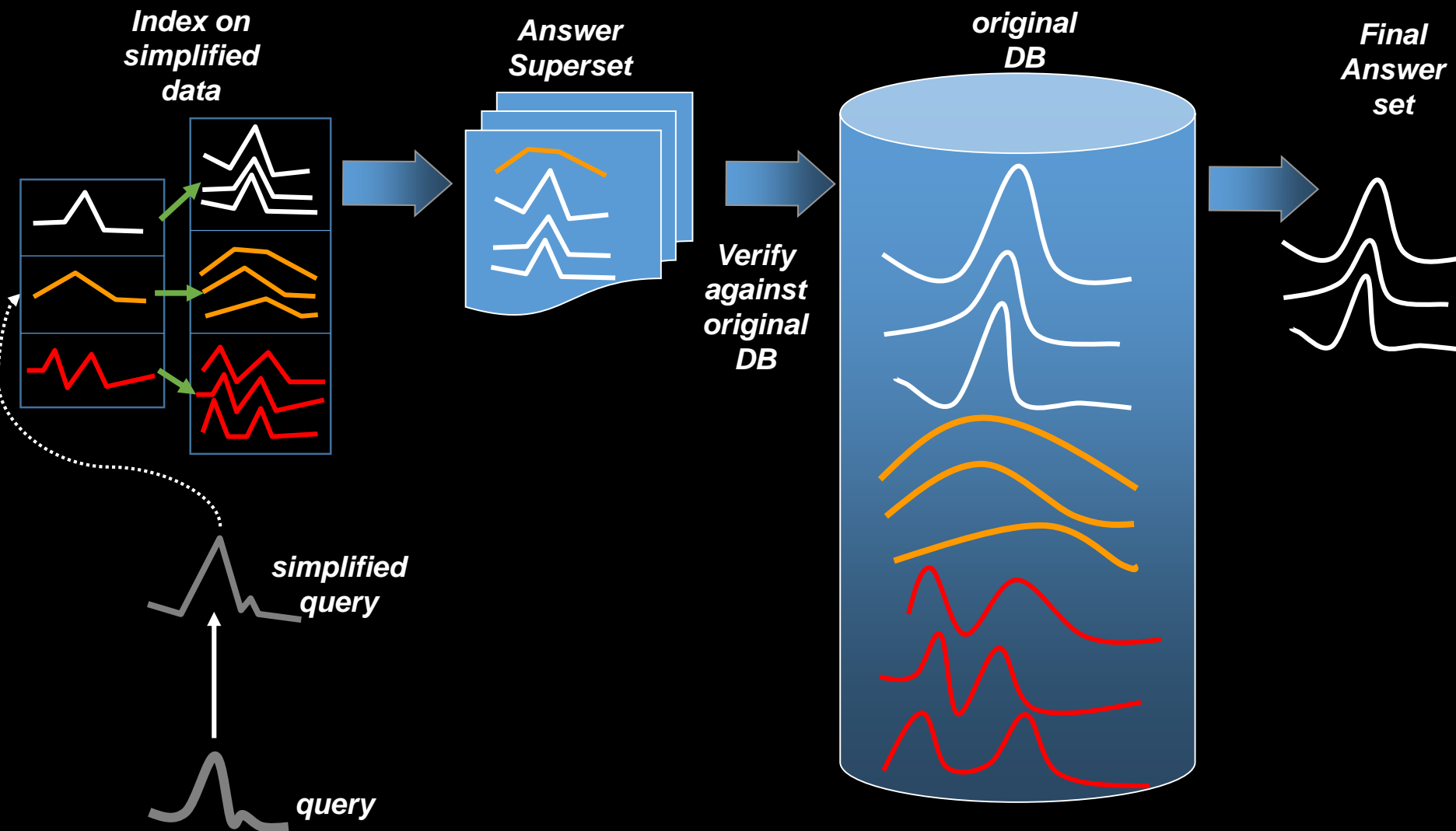
Solution: Quick-and-dirty filter:

- extract k features
- map into a point in k -d feature space
- organize points with off-the-shelf spatial access method ('SAM')
- retrieve the answer using a NN query
- discard false alarms

* R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In FODO Conference, volume 730, 1993.

C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In Proceedings 1994 ACM SIGMOD Conference, Minneapolis, MN, USA, 1994.

Generic Search



Computational cost: $O(m \log n)$

GEMINI: Key Requirement

- GEMINI works when:

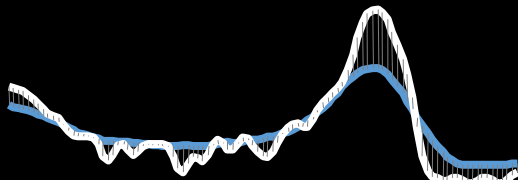
$$D_{feature}(F(x), F(y)) \leq D(x, y)$$

to ensure zero false negatives

- $D_{feature}$ is a **metric** to have working Spatial Indexes
- $D_{feature}$ is significantly **inexpensive** compared to D to gain efficiency

What is lower bounding?

Exact (Euclidean) distance $D(Q,S)$

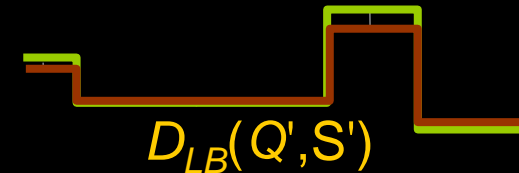
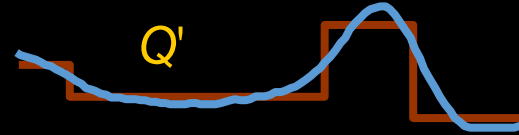


$D(Q,S)$

$D(Q,S)$

$$\equiv \sqrt{\sum_{i=1}^n (q_i - s_i)^2}$$

Lower bounding distance $D_{LB}(Q,S)$



$D_{LB}(Q',S')$

$D_{LB}(Q',S')$

$$\equiv \sqrt{\sum_{i=1}^M (q'_i - s'_i)^2}$$

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

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

Lower Bound 1: DFT Coefficients

Discrete Fourier Transform (DFT)

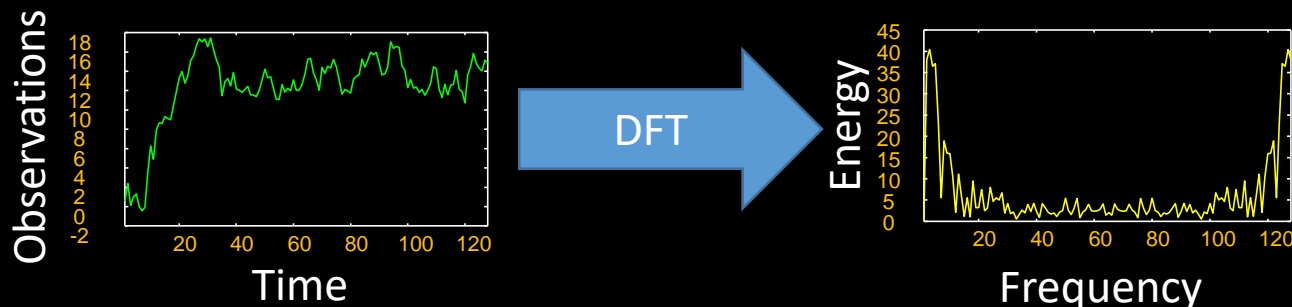
$$X_f = \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} x_t e^{-j \frac{2\pi f t}{n}} \quad j = \sqrt{-1}$$

Inverse DFT

$$x_t = \frac{1}{\sqrt{n}} \sum_{f=0}^{n-1} X_f e^{+j \frac{2\pi f t}{n}}$$

Property 1: DFT is an orthonormal transform,
 $\sum x_t^2 = \sum |X_f|^2 \rightarrow \sum (x_t - y_t)^2 = \sum |X_f - Y_f|^2$

Property 2: DFT of a real sequence is symmetric in magnitude. Thus we need only half of the transform.



Lower Bound 1:

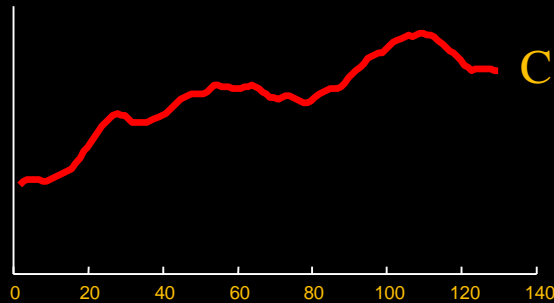
DFT based Lower Bound

$$\sum^n (x_t - y_t)^2 = 2 \sum^{n/2} |X_f - Y_f|^2 \geq 2 \sum^k |X_f - Y_f|^2$$

If we take the difference between any subset of k coefficients, we get a lower bound

Real time series contains most of the energy in few frequencies, commonly in the first few.

Lower Bound 1: Example



$$n = 128$$

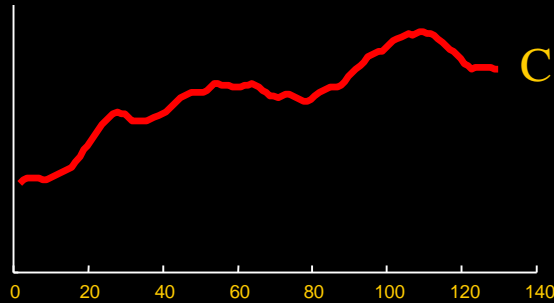
Raw Data

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
...

The graphic shows a time series with 128 points.

The raw data used to produce the graphic is also reproduced as a column of numbers (just the first 30 or so points are shown).

Lower Bound 1: Example



**Raw
Data**

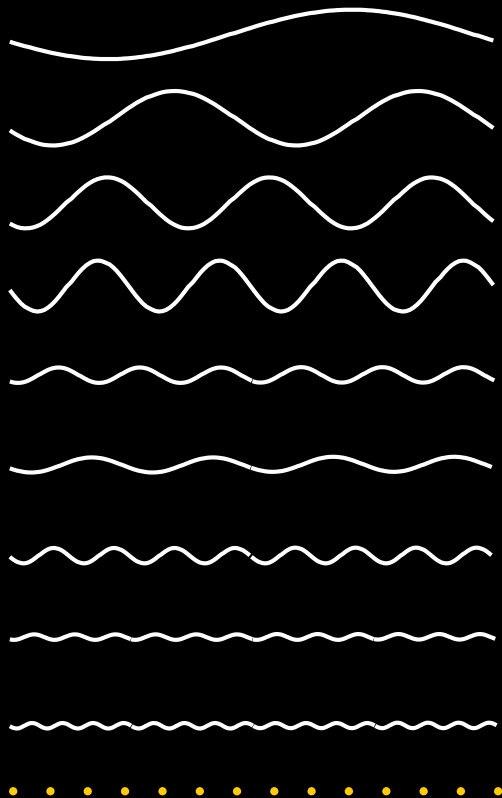
0.4995
0.5264
0.5523
0.5761
0.5973
0.6153
0.6301
0.6420
0.6515
0.6596
0.6672
0.6751
0.6843
0.6954
0.7086
0.7240
0.7412
0.7595
0.7780
0.7956
0.8115
0.8247
0.8345
0.8407
0.8431
0.8423
0.8387
...

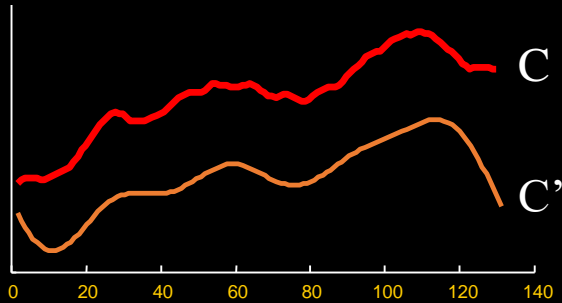
**Fourier
Coefficients**

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
...

We can decompose the data into 64 pure sine waves using the Discrete Fourier Transform (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).





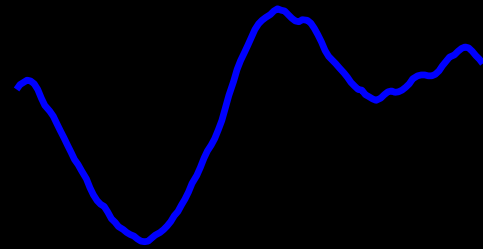
We have
discarded
of the data.

Raw Data	Fourier Coefficients	Truncated Fourier Coefficients
0.4995	1.5698	1.5698
0.5264	<u>1.0485</u>	<u>1.0485</u>
0.5523	0.7160	0.7160
0.5761	<u>0.8406</u>	<u>0.8406</u>
0.5973	0.3709	0.3709
0.6153	<u>0.4670</u>	<u>0.4670</u>
0.6301	0.2667	0.2667
0.6420	<u>0.1928</u>	<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	
...	...	

$n = 128$
 $k = 8$

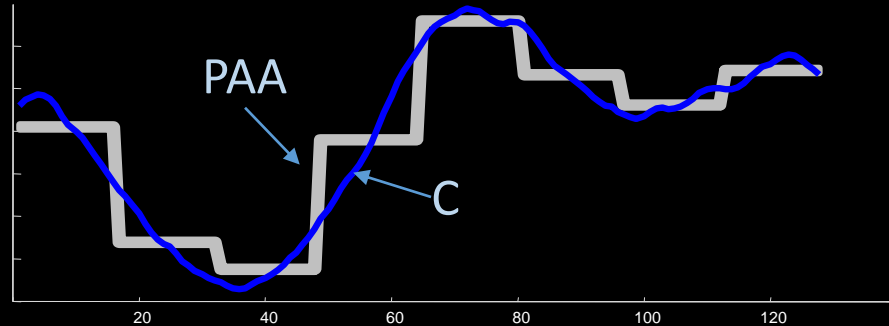
Only k coefficients can provide a good approximation of the data and produce an inexpensive lower bound.

Lower Bound 2: Symbolic Aggregate AppRoXimation (SAX)



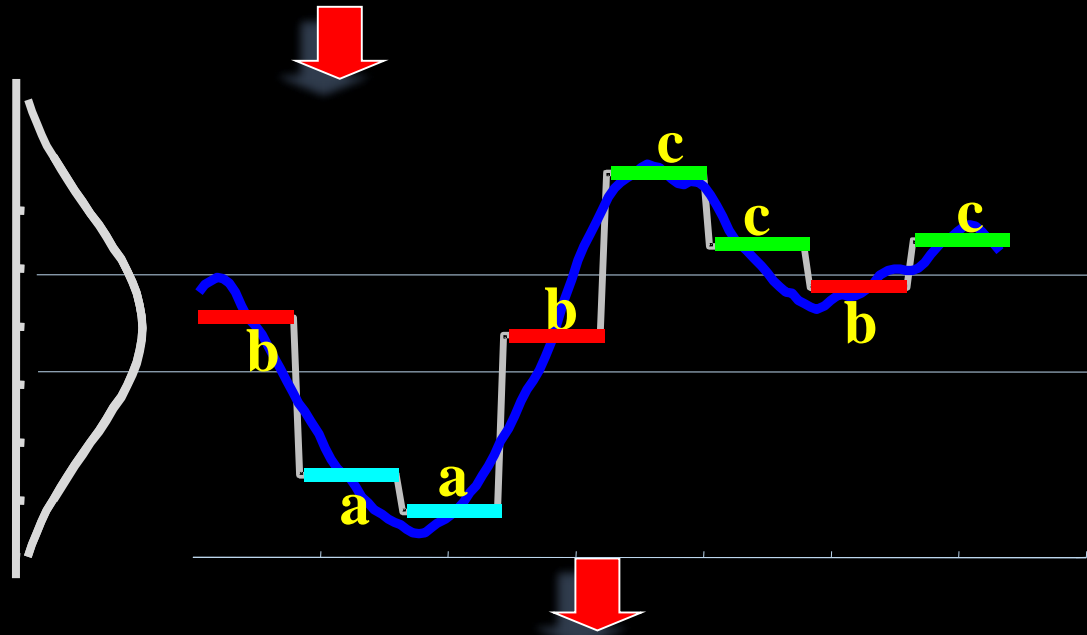
baabccbc

Lower Bound 2: How to obtain SAX



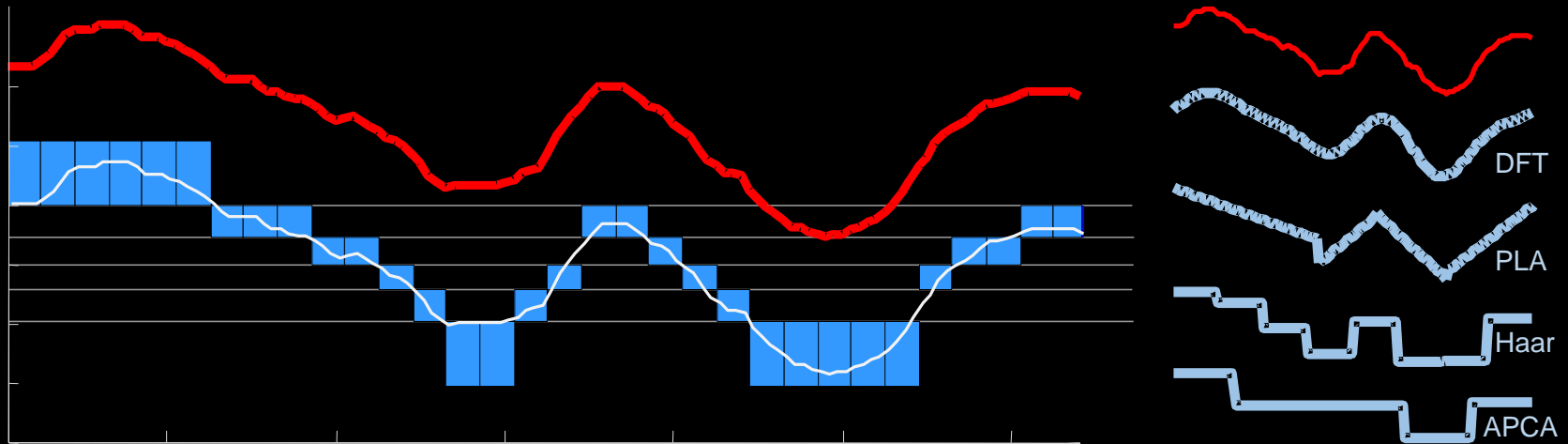
First convert the time series to PAA representation, then convert the PAA to symbols

It take linear time



baabccbc

Visual Comparison



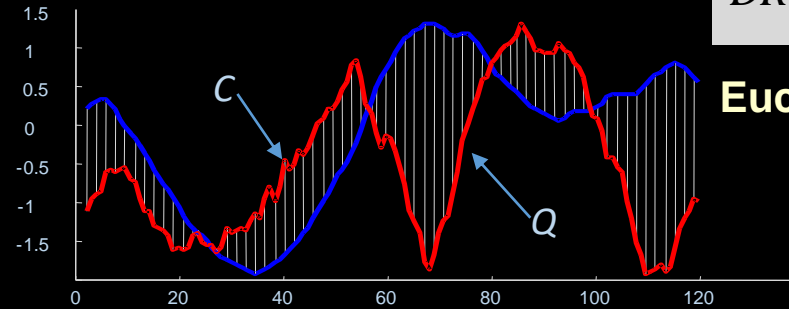
A raw time series of length 128 is transformed into the word “**ffffffeeddcbaabceedcbaaaaacddee.**”

- We can use more symbols to represent the time series since each symbol requires fewer bits than real-numbers (float, double)

Lower Bound 2:

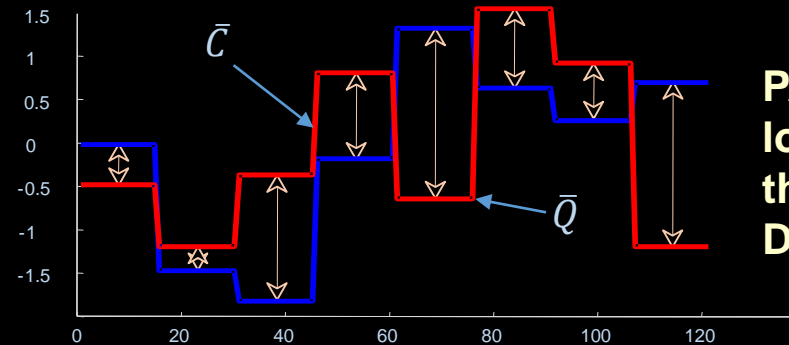
SAX provides a lower bound

$$DR(\bar{Q}, \bar{C}) \equiv \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^w (\bar{q}_i - \bar{c}_i)^2}$$



Euclidean Distance

$$DR(\bar{Q}, \bar{C}) \equiv \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^w (\bar{q}_i - \bar{c}_i)^2}$$



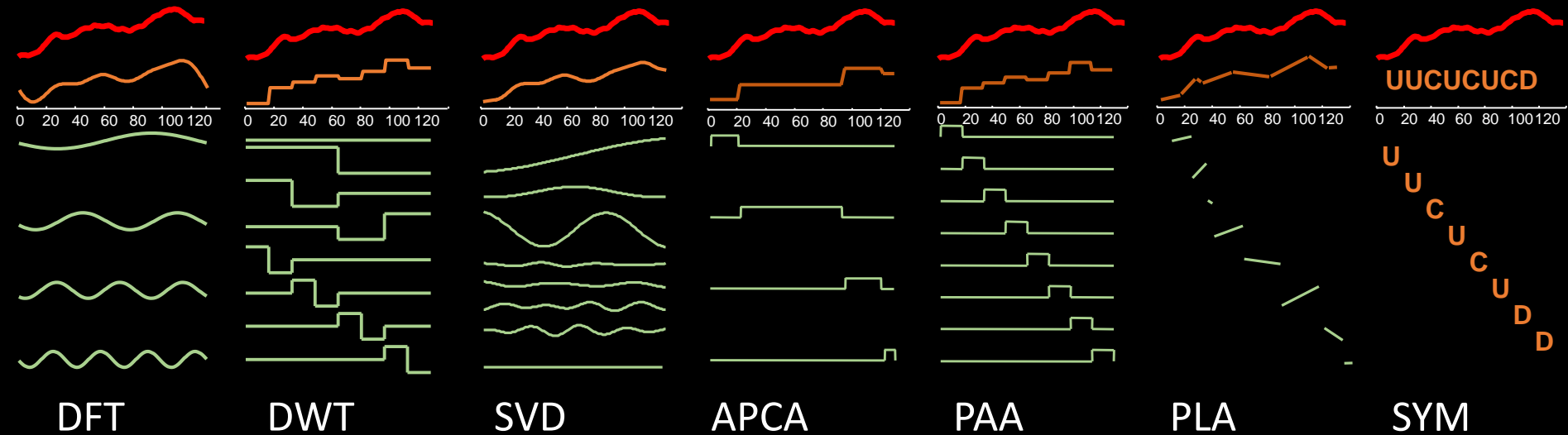
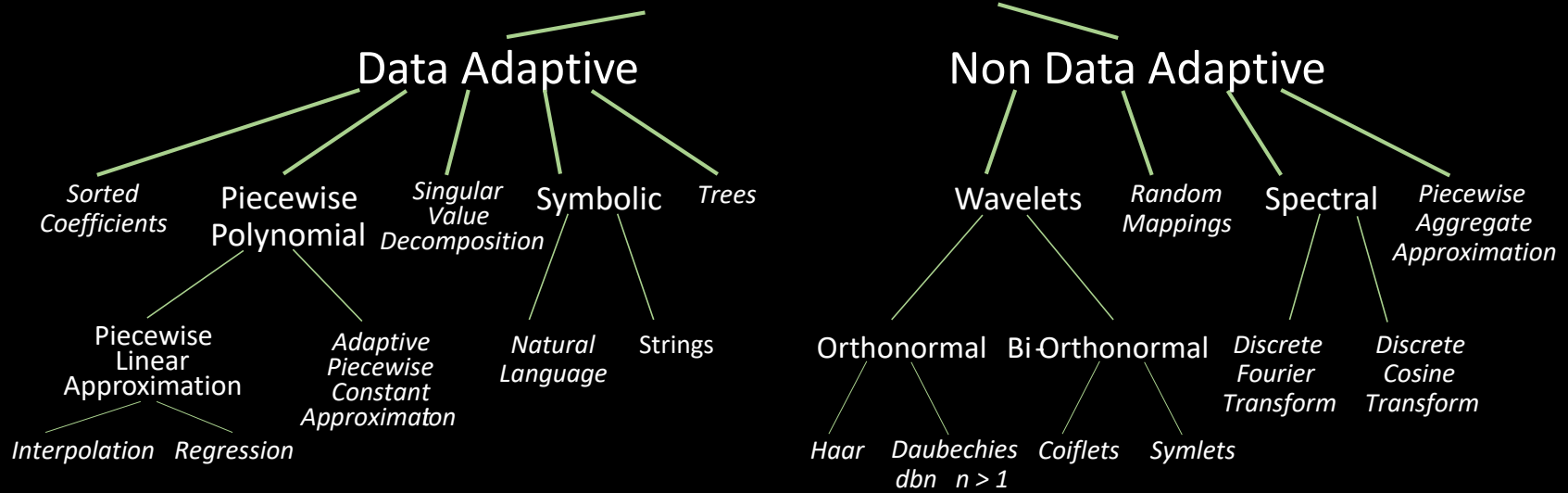
**PAA distance
lower-bounds
the Euclidean
Distance**

\hat{C} = baabccbc
 \hat{Q} = babcacca

$$MINDIST(\hat{Q}, \hat{C}) \equiv \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^w (dist(\hat{q}_i, \hat{c}_i))^2}$$

**dist() can be implemented using a
table lookup.**

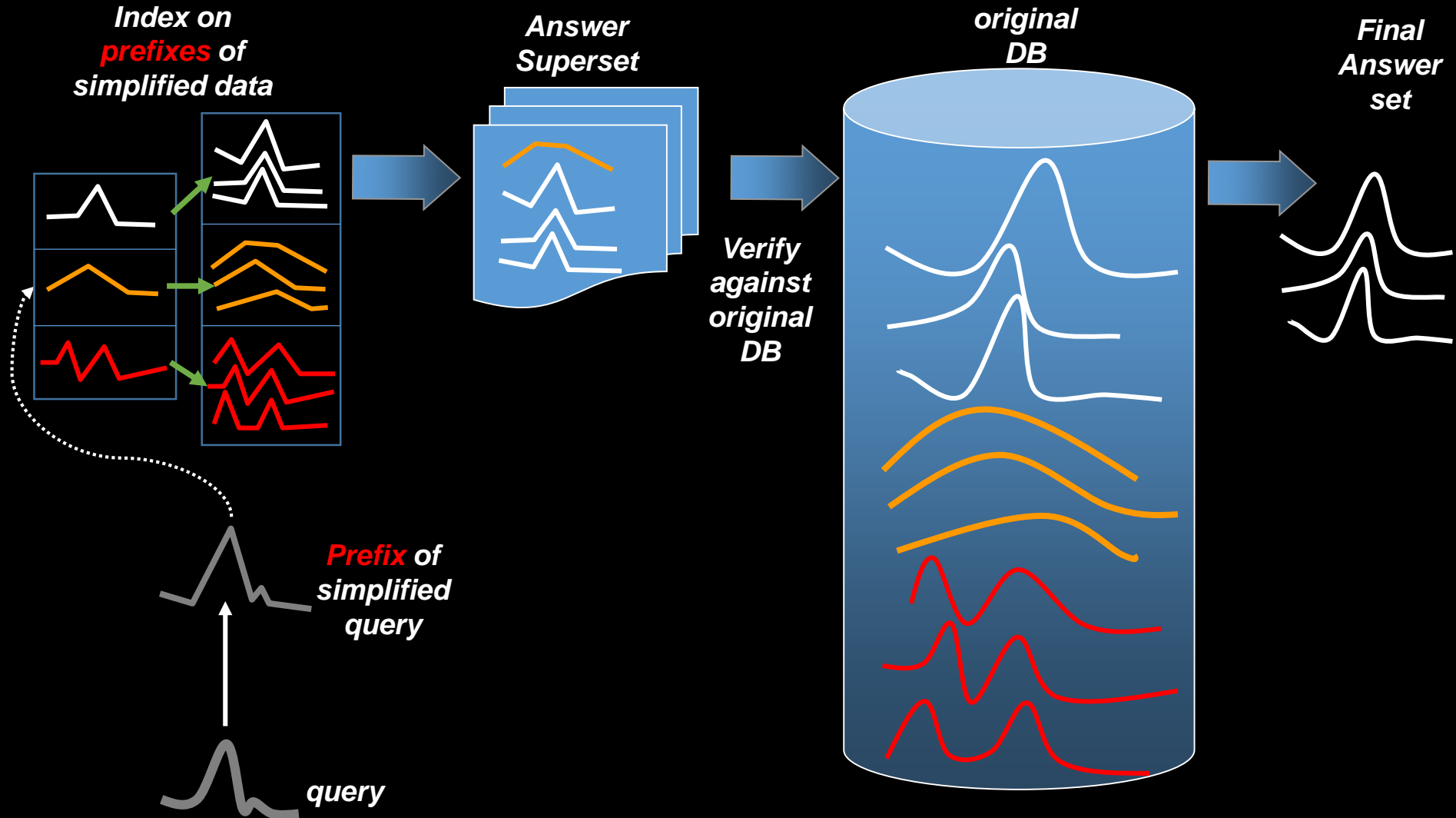
Time Series Representations



GEMINI: Other Assumptions

- The database consists of fixed length time series and the query length matches the length of the database
 - Not true for most real-application (e.g. seismology, physiology)
- The method is exact
 - Does not sell accuracy for speed
- The time series in the database are independent
 - Not true for subsequence search

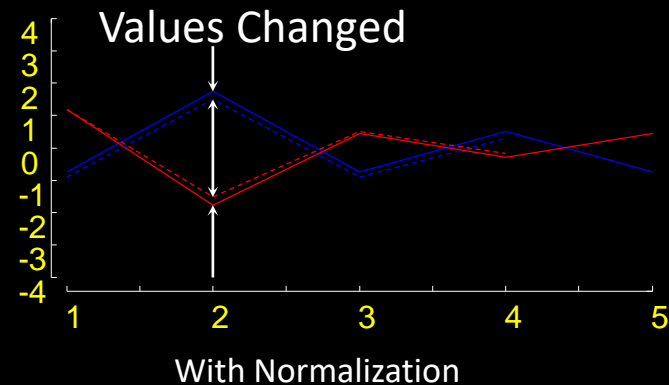
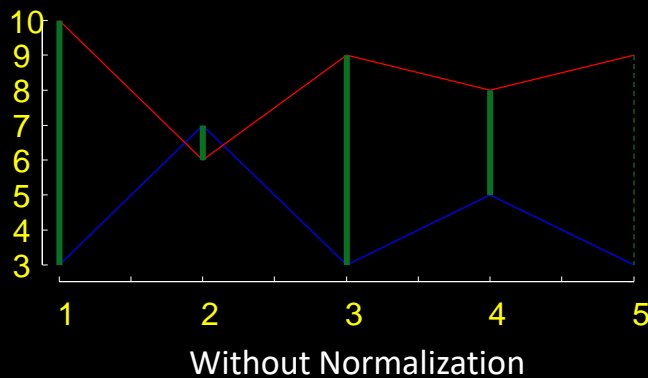
Generic Prefix-based Search



Computational cost: $O(m \log n)$

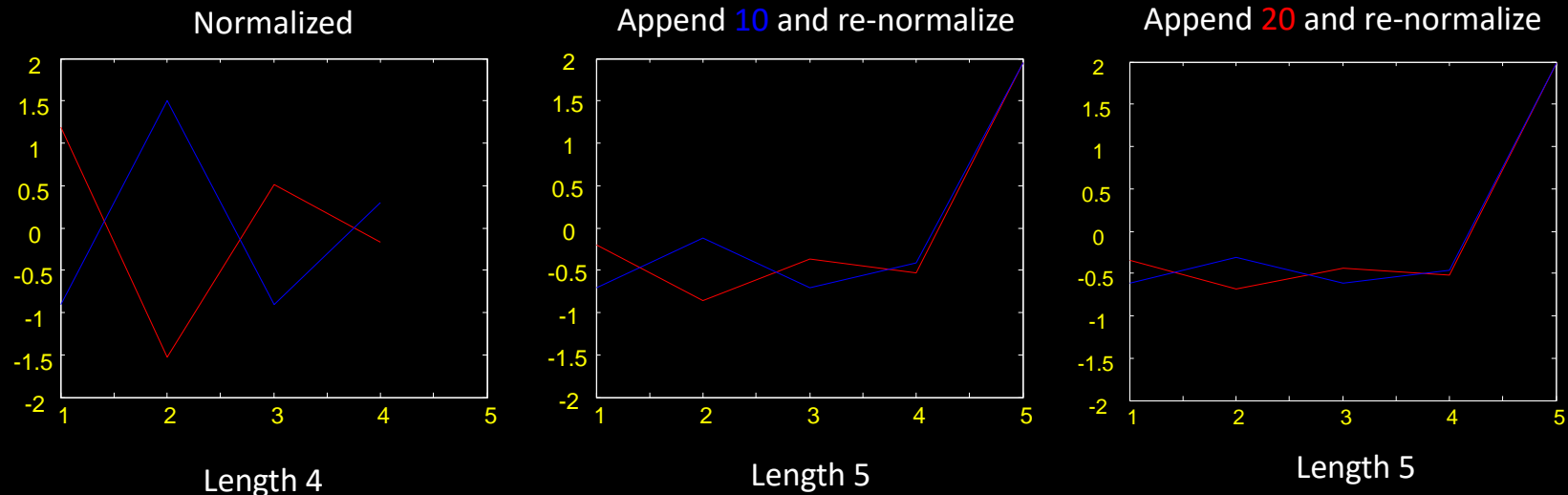
Prefix distance is not a lower bound of the original distance

1. Two time series x and y of length m
2. Prefix distance is a trivial lower bound of un-normalized distance
3. Normalized Euclidean distance $d(\hat{x}, \hat{y})$, can increase or decrease if we extend \hat{x} and \hat{y} by appending the next two numbers.



Lower-bound upon extension 1 of 3

Area between blue and red is
the distance between the signals



If infinity is appended to both the signals, they will
have zero area/distance.

Lower-bound upon extension 2 of 3

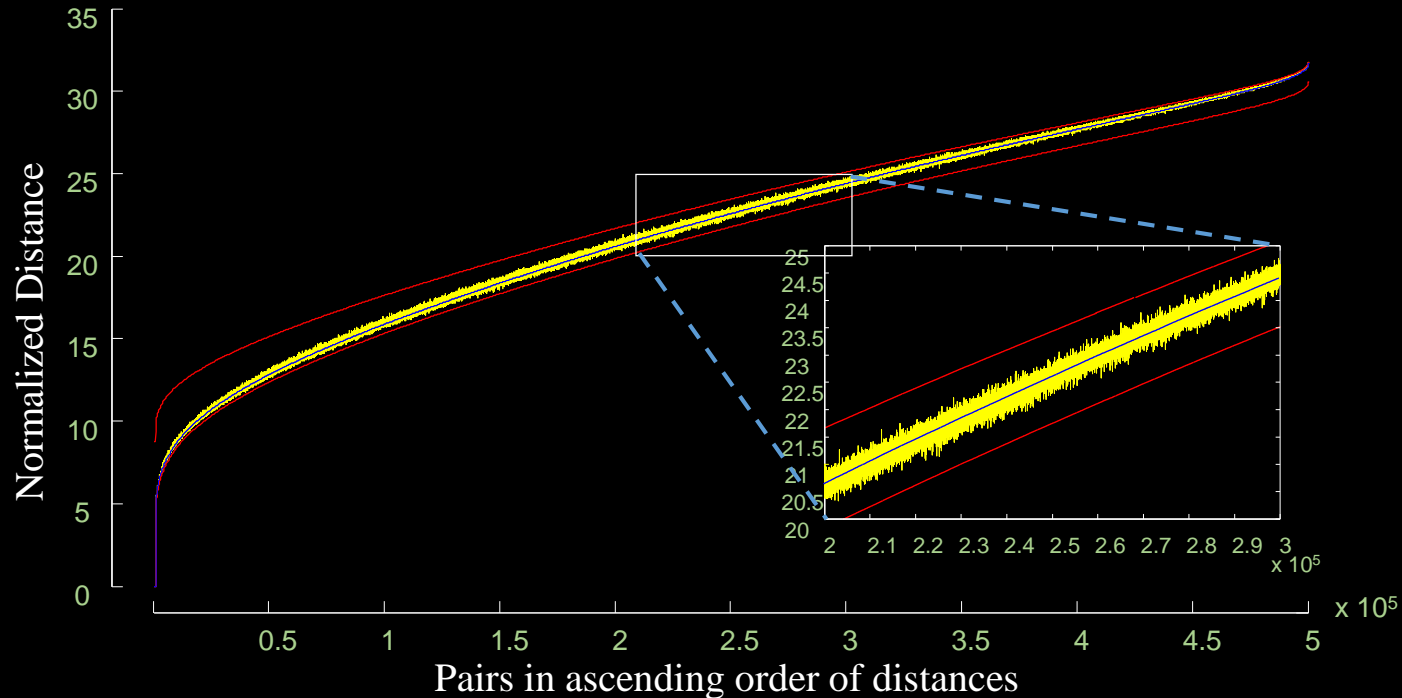
$$\bullet d_{LB}^2(\hat{\mathbf{x}}_{+1}, \hat{\mathbf{y}}_{+1}) = \frac{1}{\sigma_m^2} d_m^2(\hat{\mathbf{x}}, \hat{\mathbf{y}}) < d_m^2(\hat{\mathbf{x}}, \hat{\mathbf{y}})$$

$$\text{Variances of } \hat{\mathbf{x}}_{+1} \text{ and } \hat{\mathbf{y}}_{+1}, \sigma_m^2 = \frac{m}{m+1} + \frac{m}{(m+1)^2} z^2$$

z = maximum normalized value in the database

A safe approximation of $z = \max(\text{abs}(\hat{\mathbf{x}}), \text{abs}(\hat{\mathbf{y}}))$

Lower-bound upon extension 3 of 3

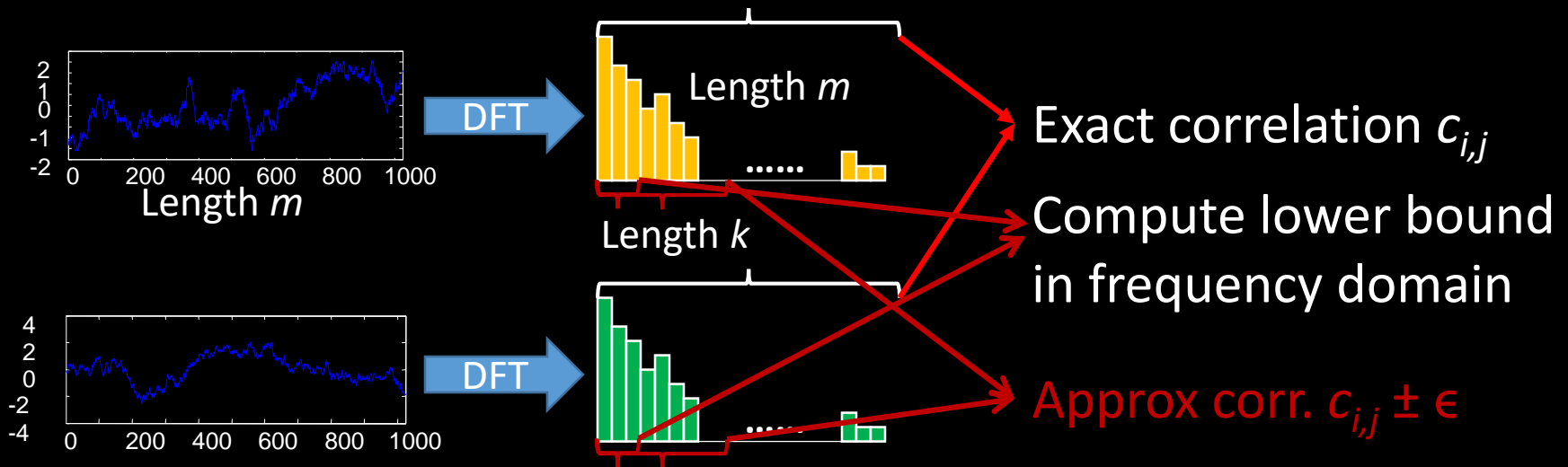


Blue: Distances of random signals of length 255

Yellow: Distances of the signals when they are extended by one random sample

Red: The upper and lower bounds before observing the extensions

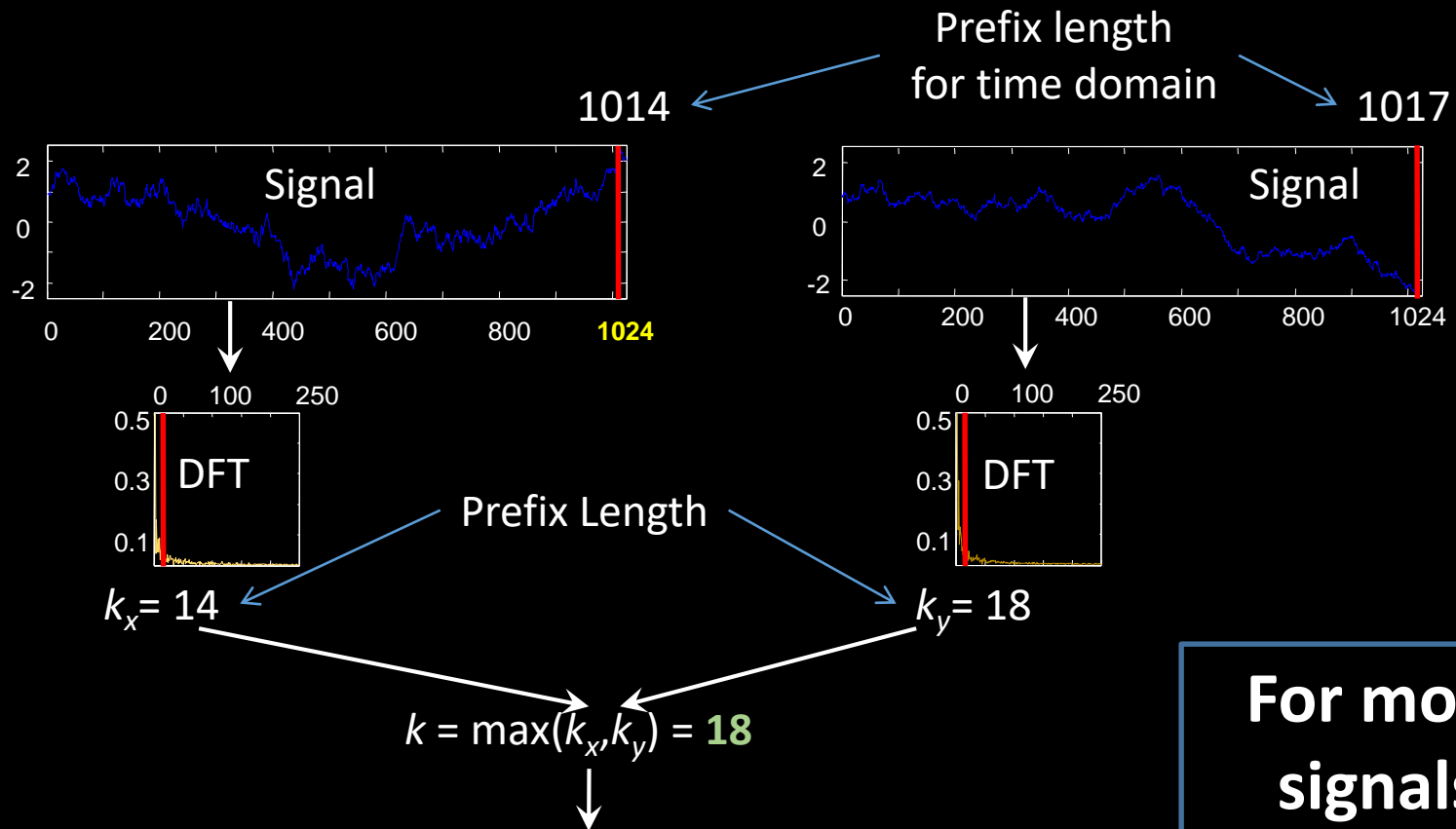
ϵ - Approximate Correlation(1)



- **Result:** given ϵ , compute k
- Smooth Signal $\Rightarrow k$ is small for small ϵ

Reduce scanning cost from $O(m)$ to $O(k)$

Example: $\epsilon=0.1$



**For most
signals**

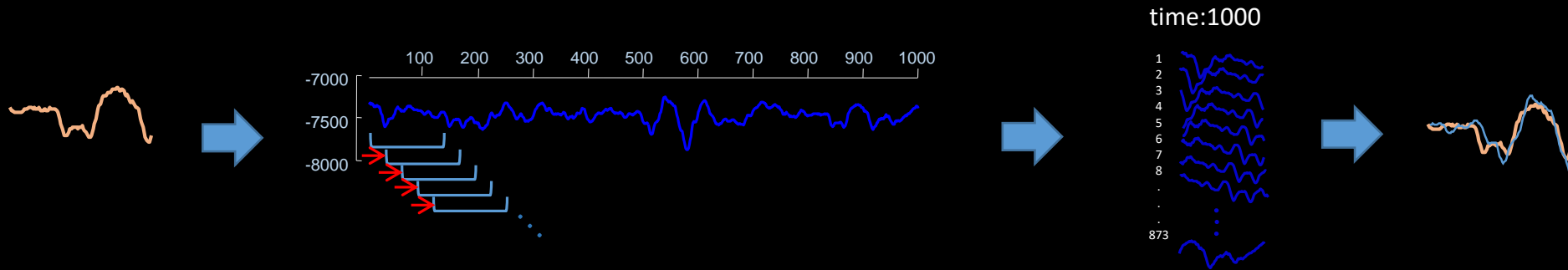
$$k \ll m$$

$$\text{corr}(x, y) - 0.1 < 1 - \sum_i^k (|X_i - Y_i|)^2 < \text{corr}(x, y) + 0.1$$

GEMINI: Other Assumptions

- The database consists of fixed length time series and the query length matches the length of the database
 - Not true for most real-application (e.g. seismology, physiology)
- The method is exact
 - Does not sell accuracy for speed
- **The time series in the database are independent**
 - Not true for subsequence search

Subsequence Similarity Search



- Subsequences have large overlaps that GEMINI does not exploit
- Normalization makes it difficult to exploit the overlaps
- Two techniques to exploit overlaps
 - **just-in-time normalization** with constant overhead per comparison
 - **MASS: Mueen's Algorithm for Similarity Search** using convolution

Just-in-time Normalization

- In one pass, calculate cumulative sums of over x and x^2 and store
$$C = \sum x \quad C^2 = \sum x^2$$
- Subtract two cumulative sums to obtain the sum over a window
$$S_i = C_{i+w} - C_i \quad S_i^2 = C_{i+w}^2 - C_i^2$$
- Use the sums to calculate the means and standard deviations of all windows in linear time

$$\mu_i = \frac{S_i}{w} \quad \sigma_i = \sqrt{\frac{S_i^2}{w} - \left(\frac{S_i}{w}\right)^2}$$

- Dynamically normalize observations when calculating distance and possibly abandon early

$$cost = \left(\frac{x_{ij} - \mu_{xi}}{\sigma_{xi}} - \frac{q_i - \mu_{qi}}{\sigma_{qi}} \right)^2$$

Early Abandoning

Given two time series

$$\mathbf{x} = x_1 \dots x_n$$

and

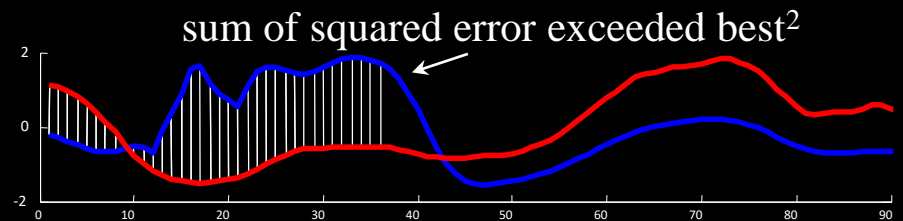
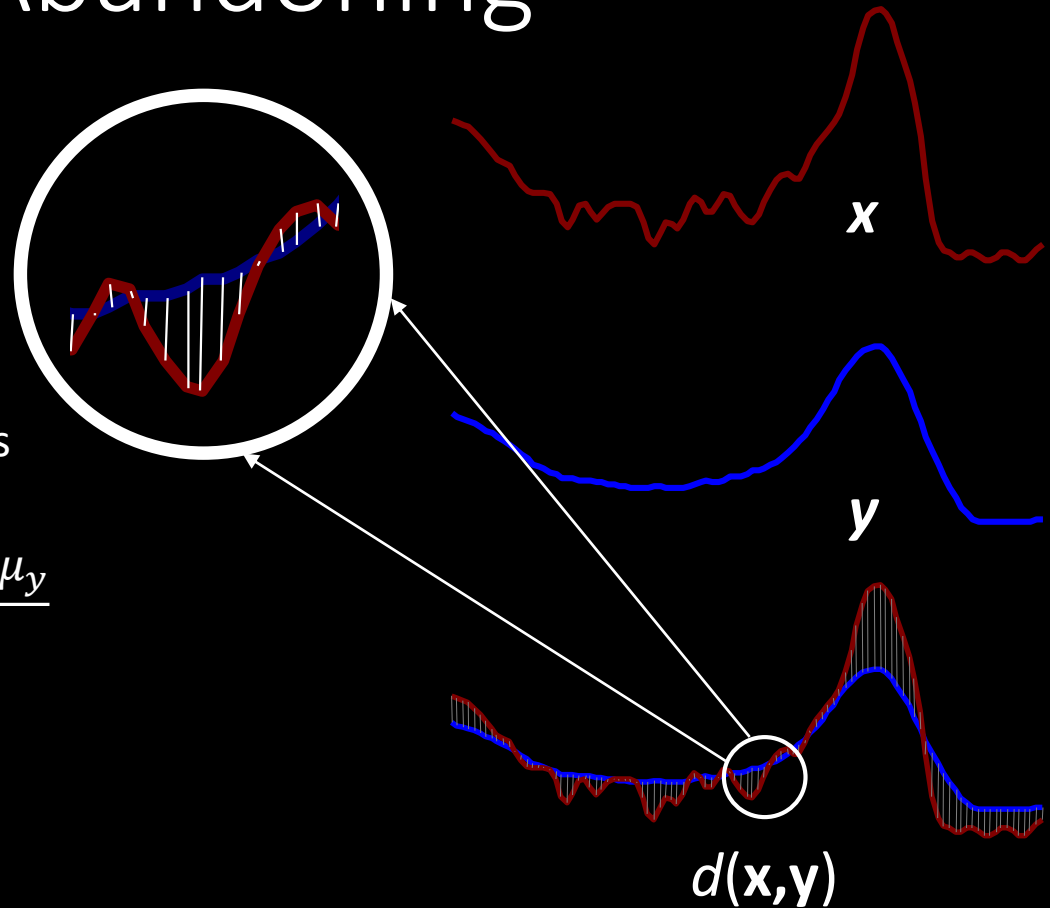
$$\mathbf{y} = y_1 \dots y_n$$

their z-Normalized Euclidean distance is defined as:

$$\hat{x}_i = \frac{x_i - \mu_x}{\sigma_x} \quad \hat{y}_i = \frac{y_i - \mu_y}{\sigma_y}$$

$$d(x, y) = \sqrt{\sum_{i=1}^n (\hat{x}_i - \hat{y}_i)^2}$$

Early abandoning reduces number of operations when minimizing



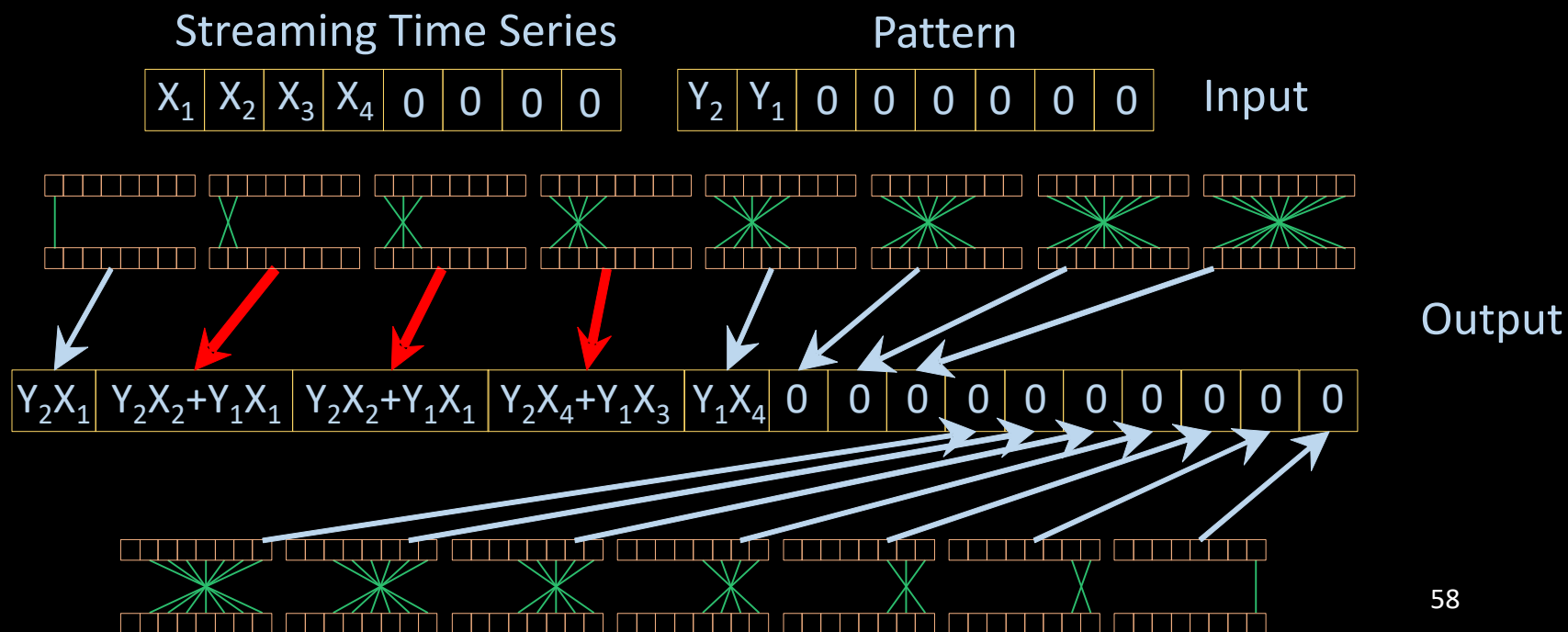
Mueen's Algorithm for Similarity Search (MASS) (1 of 4)

- Convolution based method
 - $O(n \log n)$ cost to obtain all the sliding dot products
- On-the-fly normalization
 - Use the dot products to calculate normalized distances in $O(n)$ cost

Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, Eamonn Keogh (2016). Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View that Includes Motifs, Discords and Shapelets. IEEE ICDM 2016.

Mueen's Algorithm for Similarity Search (MASS) (2 of 4)

- Double the time series by appending zeros
- Reverse the normalized query and append zeros to match length
- Use FFT based convolution technique
 - $\text{ifft}(\text{fft}(x) \cdot \text{fft}(y))$



Mueen's Algorithm for Similarity Search (MASS) (3 of 4)

- Since query is normalized, we reform the distance function by applying $\sum y = 0, \sum y^2 = n$

- $$d^2(x, y) = \sum \left(\frac{(x - \mu_x)}{\sigma_x} - y \right)^2$$
$$= \frac{\sum x^2 - 2 \sum x \mu_x + n \mu_x^2}{\sigma_x^2} - 2 \frac{\sum xy}{\sigma_x} + n$$

- The distance can be calculated by using the sliding dot products and cumulative sums

Mueen's Algorithm for Similarity Search (MASS) (4 of 4)

- Produces a “distance profile” of the query to the subsequences of the time series. Every distance is reported, nothing is abandoned
- Can be used to answer K-NN queries, range queries, and density estimation all at the same time
- Data and query independent execution.
- Can be further optimized when multiple queries are issued



Top-level Outline

- Similarity Measures
- Similarity Search
 - Lock-step search (e.g. Correlation Search)
 - Elastic search (e.g. DTW search)

Similarity Search under DTW

- The general conception: *DTW is slow and we have a never-ending need for speed*
 - Better performance in knowledge extraction
 - Better scalability to process BigData
 - Better interactivity in human driven data analysis

What can be made fast?

- **One-to-One comparison**
 - Exact Implementation and Constraints
 - Efficient Approximation
 - Exploiting Sparsity
- **One-to-Many comparisons**
 - Nearest Neighbor Search
 - In subsequences of a long time series
 - Density Estimation
 - In clustering
 - Averaging Under Warping
 - In classification
- **Many-to-Many comparisons**
 - All-pair Distance Calculations

Speeding up DTW: one-to-one

- **One-to-One comparison**
 - Exact Implementation
 - Efficient Constraints
 - Exploiting Hardware
 - Efficient Approximation
 - Exploiting Sparsity

Simplest Exact Implementation

Input: x and y are time series of length n and m

Output: DTW distance d between x and y

```
D(1:n+1,1:m+1) = inf;
```

```
D(1,1) = 0;
```

```
for i = 2 : n+1 %for each row
```

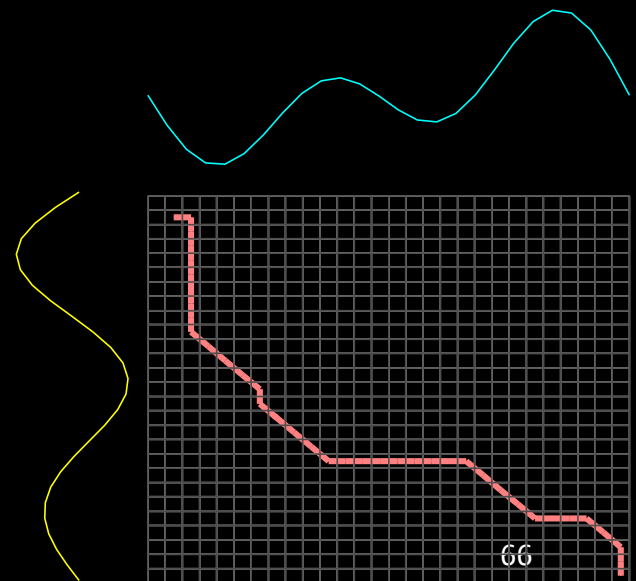
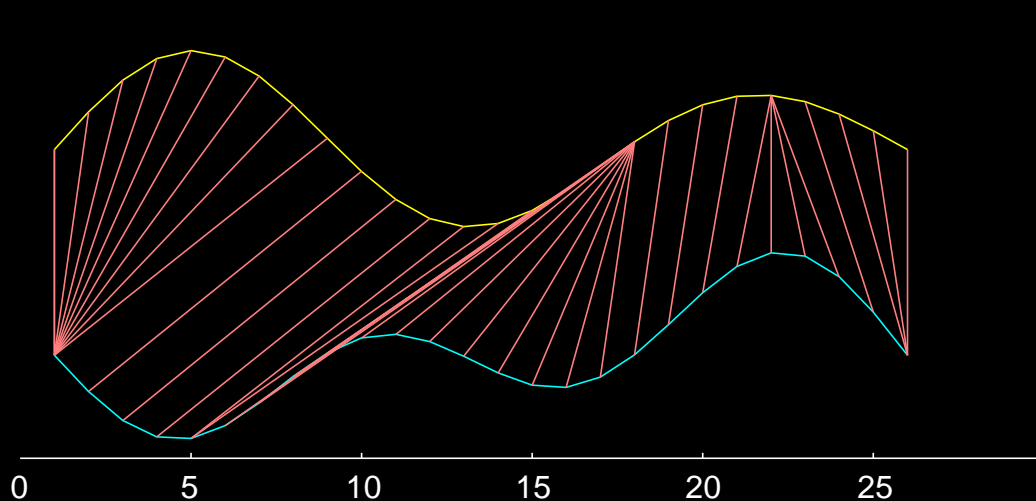
```
    for j = 2 : m+1 %for each column
```

```
        cost = (x(i-1)-y(j-1))^2;
```

```
        D(i,j) = cost + min( [ D(i-1,j), D(i,j-1), D(i-1,j-1) ] );
```

```
d = sqrt(D(n+1,m+1));
```

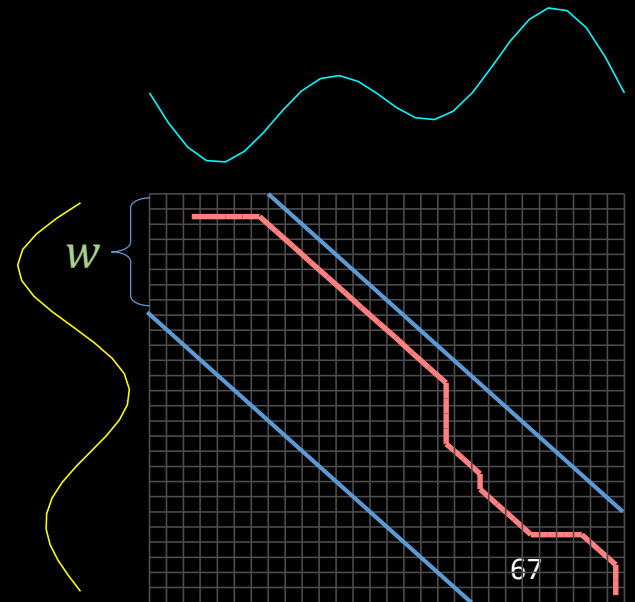
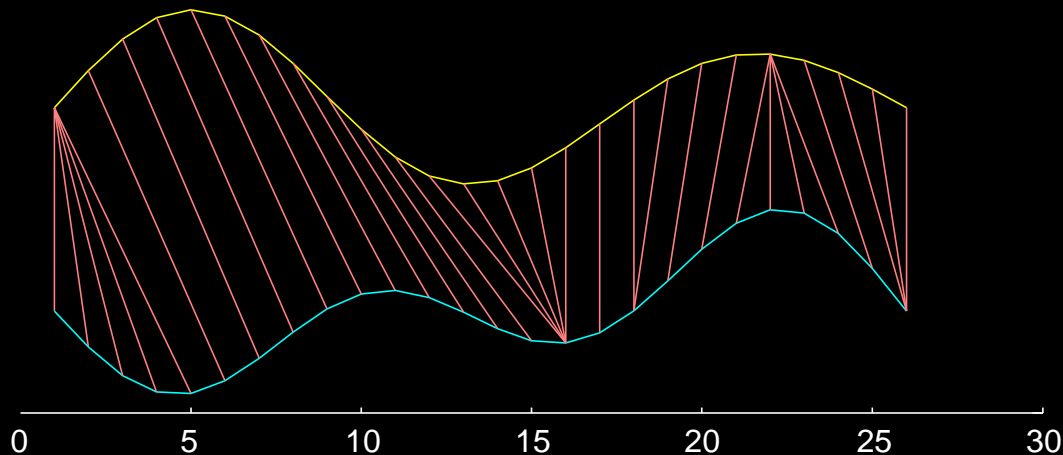
$O(n^2)$ time
 $O(n^2)$ space



Simplest Implementation (Constrained)

```
D(1:n+1,1:m+1) = inf;  
D(1,1) = 0;  
w = max(w, abs(n-m));  
for i = 2 : n+1  
    for j = max(2,i-w) : min(m+1,i+w)  
        cost = (x(i-1)-y(j-1))^2;  
        D(i,j) = cost + min( [ D(i-1,j), D(i,j-1), D(i-1,j-1) ] );  
    end  
end  
d = sqrt(D(n+1,m+1)); 6767
```

$O(nw)$ time
 $O(n^2)$ space



Memoization

```

D(2,1:m+1) = inf;
D(1,1) = 0;
p = 1; c = 2;
for i = 2 : n+1
    for j = 2 : m+1
        cost = (x(i-1)-y(j-1))^2;
        D(c,j) = cost + min( [ D(p,j), D(c,j-1), D(p,j-1) ] );
        swap(c,p);
    end
end
d = sqrt(D(n+1,m+1));

```

$O(n^2)$ time

$O(n)$ space

Previous Row



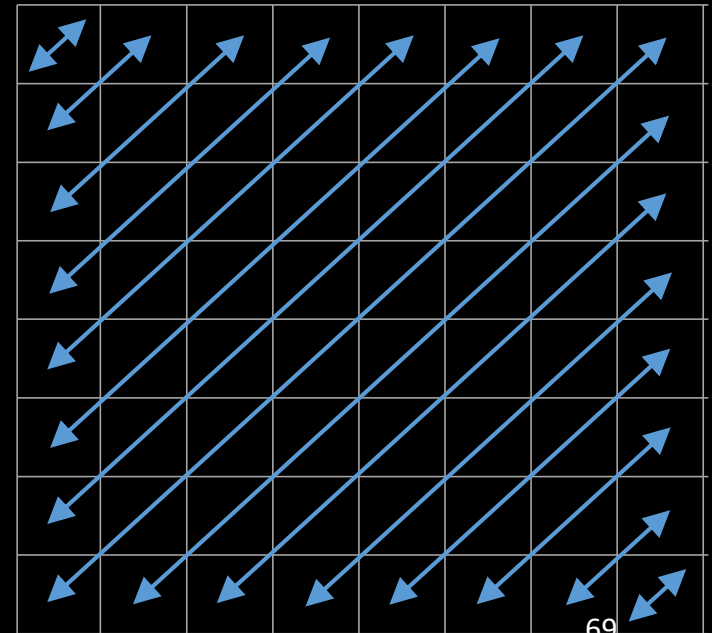
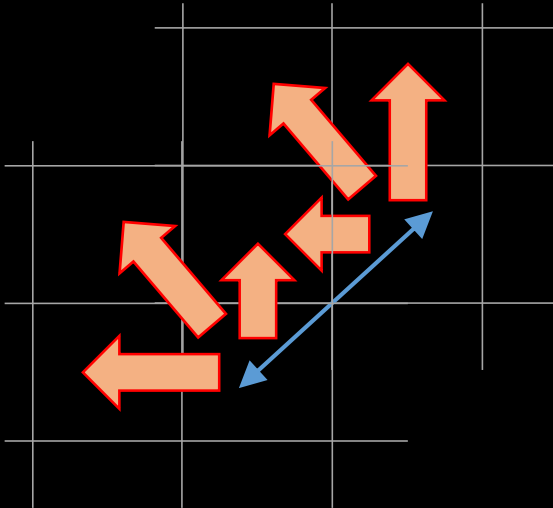
∞	1	2	2	1	2	3	4	5	3	2	1
∞	1	3	3								

Current Row

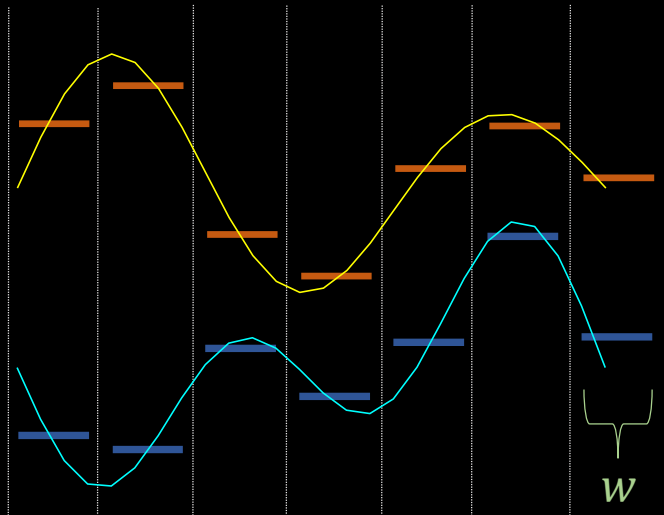
Hardware Acceleration

- Single Instruction Multiple Data (SIMD) architecture
- Cells on a diagonal are computed in parallel
- Values of a diagonal depend on the previous two diagonals

$O(n)$ time
 $O(n)$ space

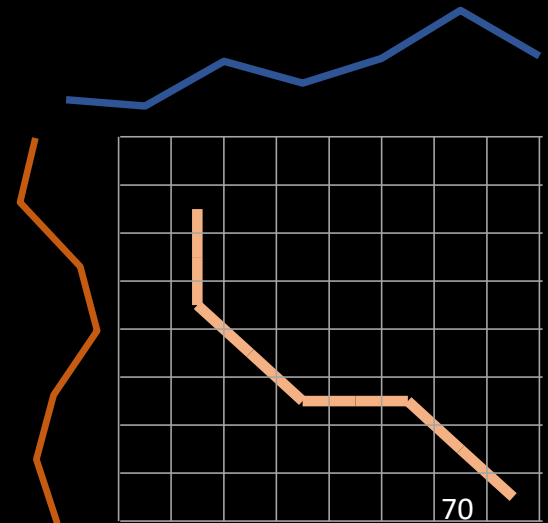
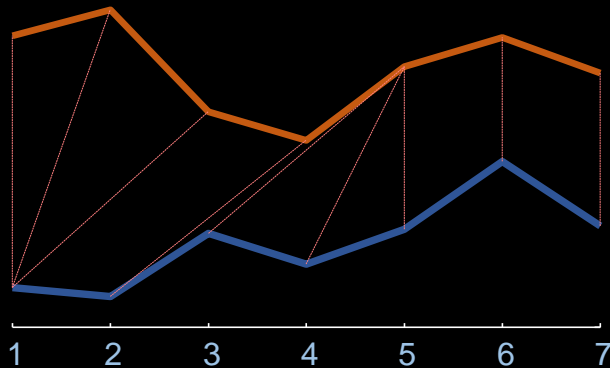


PAA based approximation



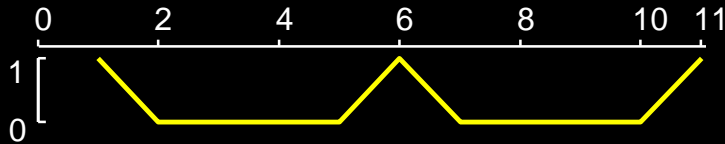
$$O\left(\frac{n}{w}\right)^2 \text{ time}$$
$$O\left(\frac{n}{w}\right)^2 \text{ space}$$

Piecewise Aggregate Approximation

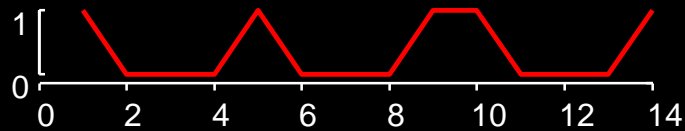


Approximation by Length-encoding

To exploit sparsity,
encode lengths of the
runs of zeros



1 0 0 0 0 1 0 0 0 0 1

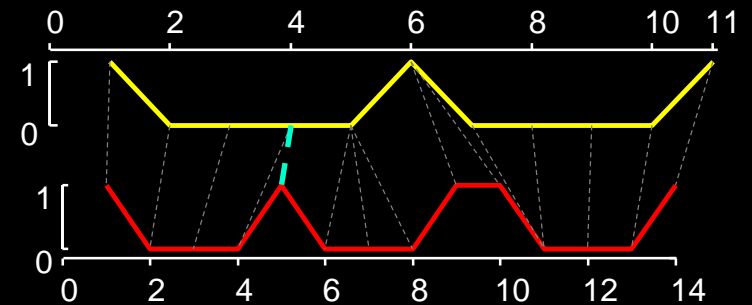
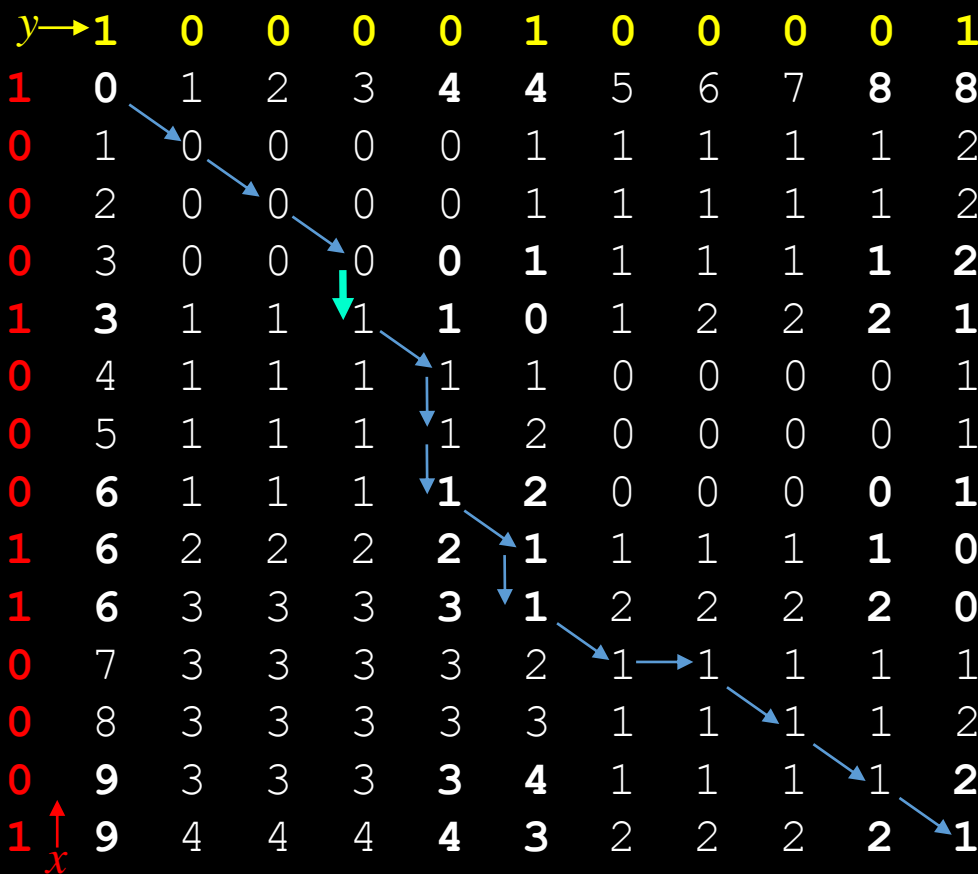


1 0 0 0 1 0 0 0 1 1 0 0 0 1

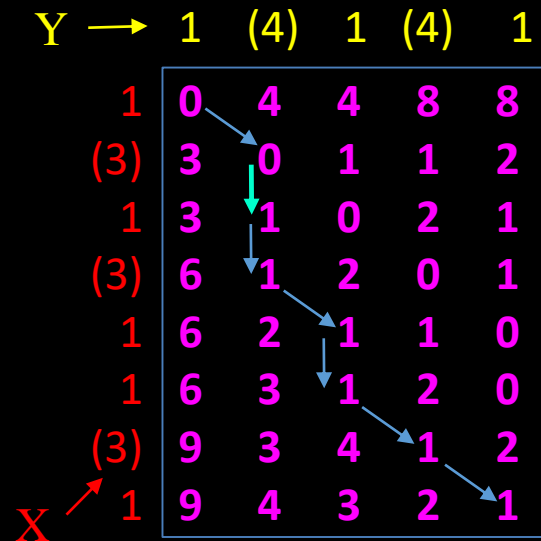
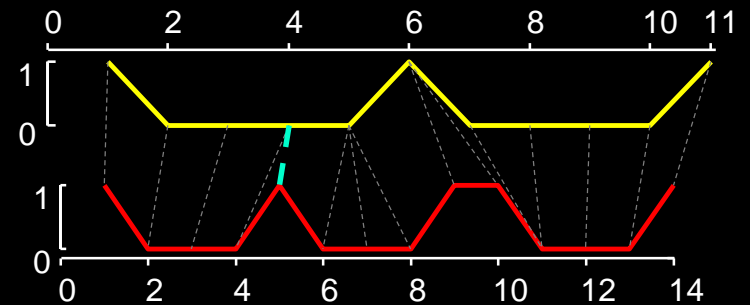
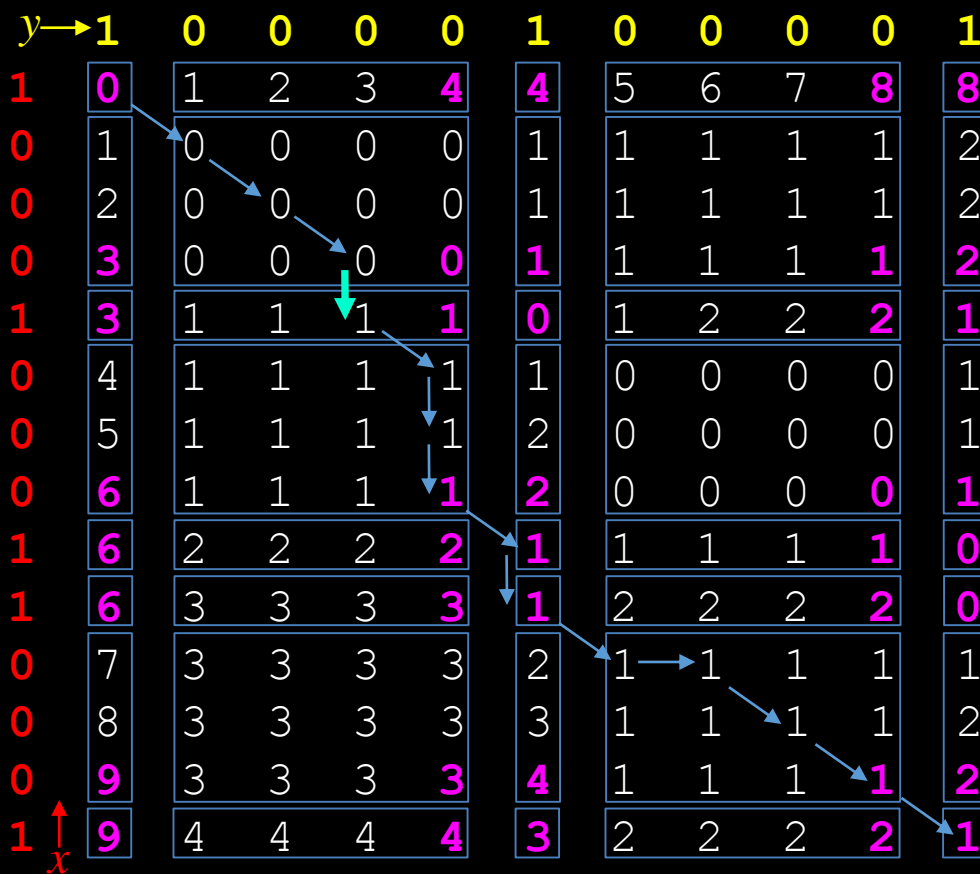
1 (4) 1 (4) 1

1 (3) 1 (3) 1 1 (3) 1

Exploiting Sparsity (1)

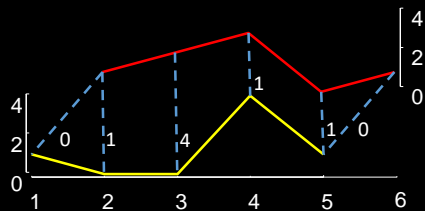


Exploiting Sparsity (2)

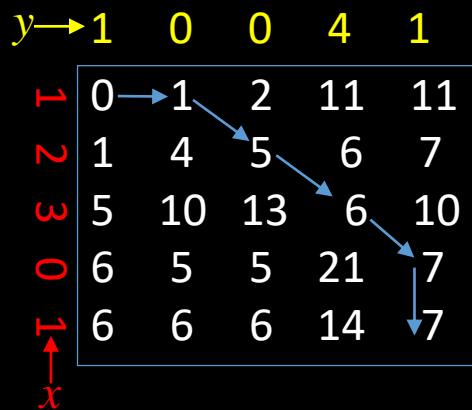


Exploiting Sparsity (2)

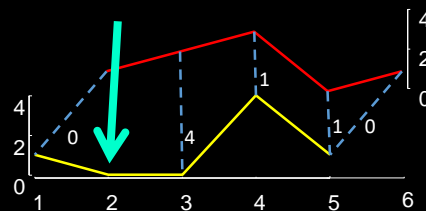
Correct Alignment



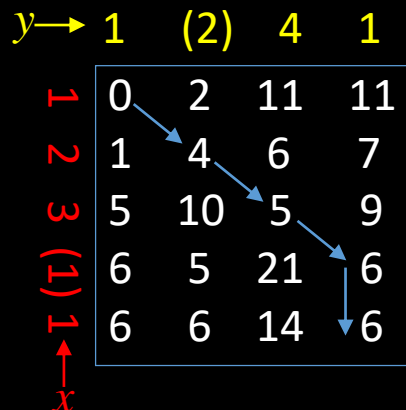
Sub-Linear change
→ Exact Distance



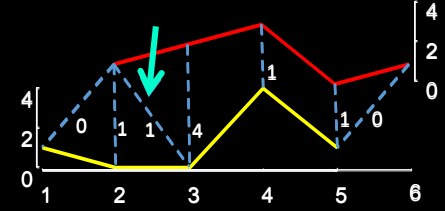
Missing Alignment



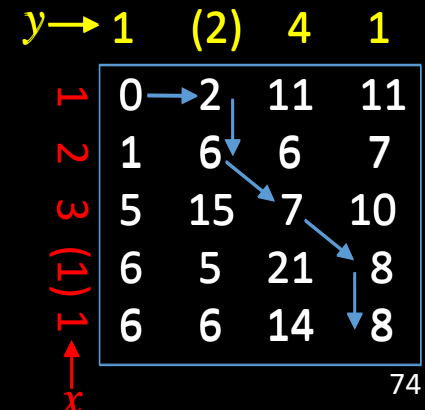
No change
→ Lower bound



Extra Alignment

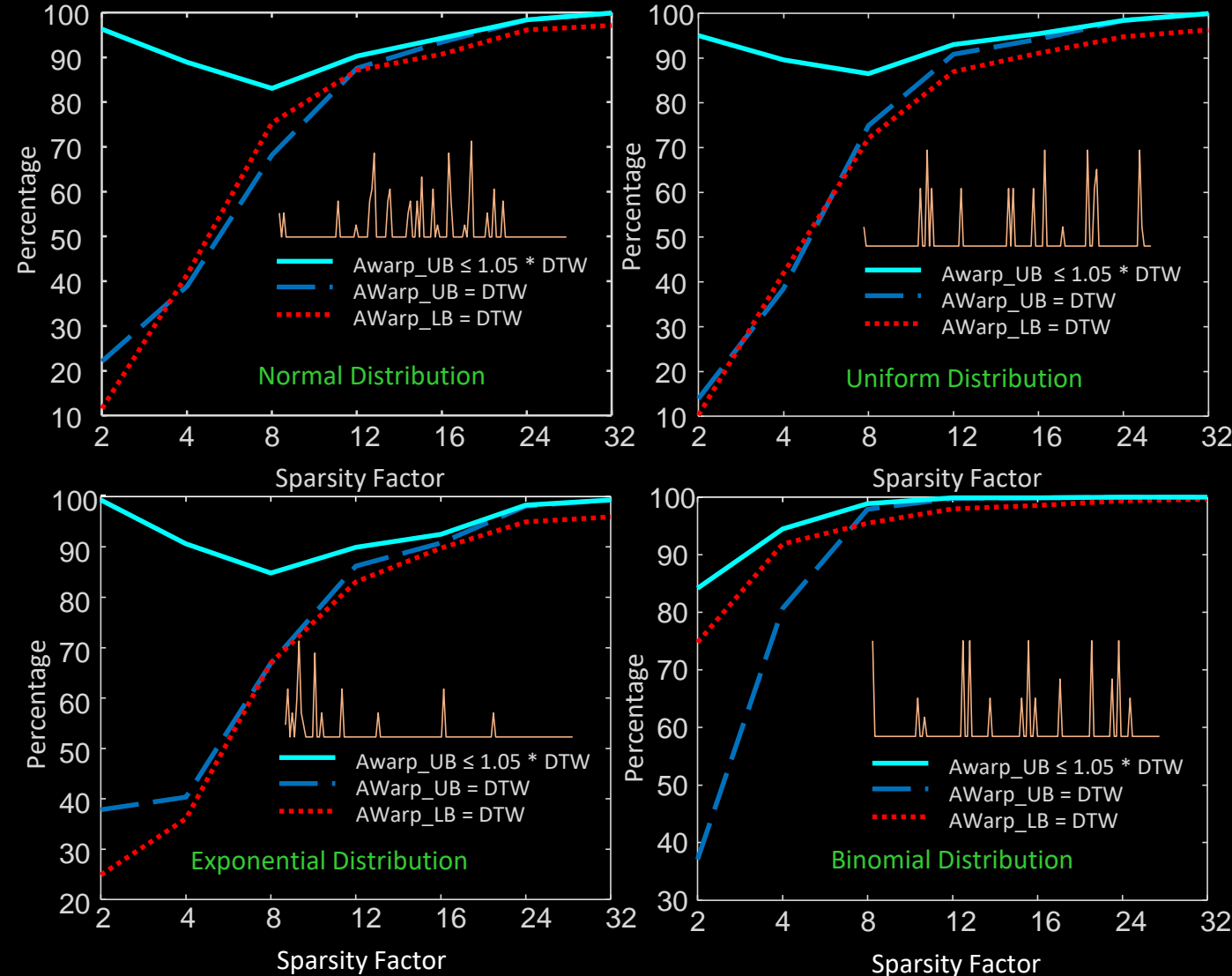


Linear change
→ Upper bound



Exploiting Sparsity (3)

Sparsity Factor of s
means $\frac{1}{s}\%$ of the time
series is filled with non-
zeros.



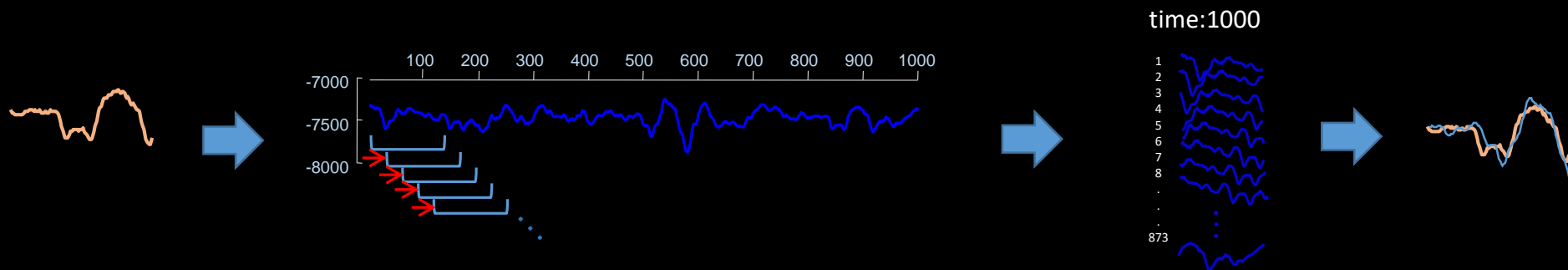
What can be made fast?

- One-to-One comparison
 - Exact Implementation and Constraints
 - Efficient Approximation
 - Exploiting Sparsity
- **One-to-Many comparisons**
 - Nearest Neighbor Search
 - In subsequences of a long time series
 - Density Estimation
 - In clustering
 - Averaging Under Warping
 - In classification
- Many-to-Many comparisons
 - All-pair Distance Calculations

Nearest Subsequence Search

- A **Q** query is given
- A long time series of length n
- $O(n)$ distance calculations are performed to

Find THE nearest subsequence of the given query under DTW.



Brute Force Subsequence Search

Computational cost: $O(nm^2)$

Algorithm Sequential_Scan(Q)

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

Lower Bounding Nearest Neighbor Search

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.



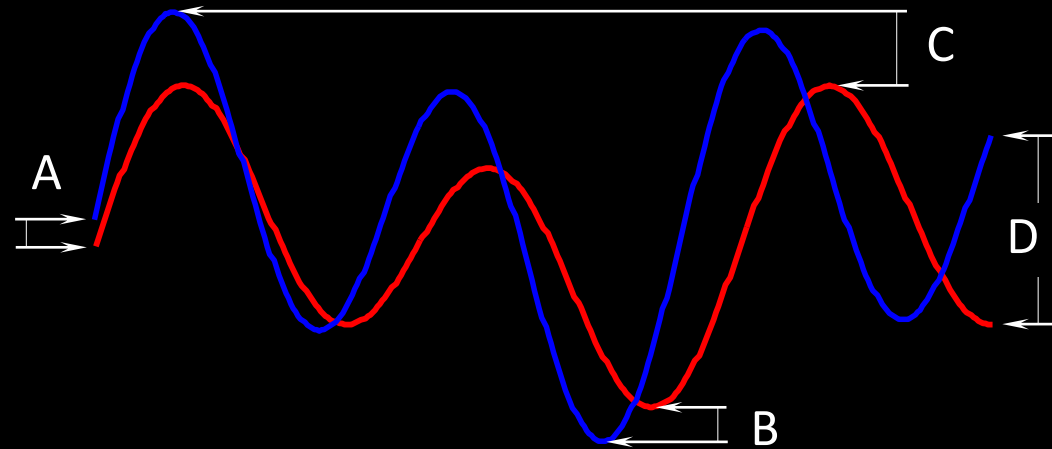
Only do the expensive, full calculations when it is absolutely necessary



Lower Bound of Kim



LB_Kim



$O(1)$ time if considered only first and last points

$O(n)$ time for all four distances

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

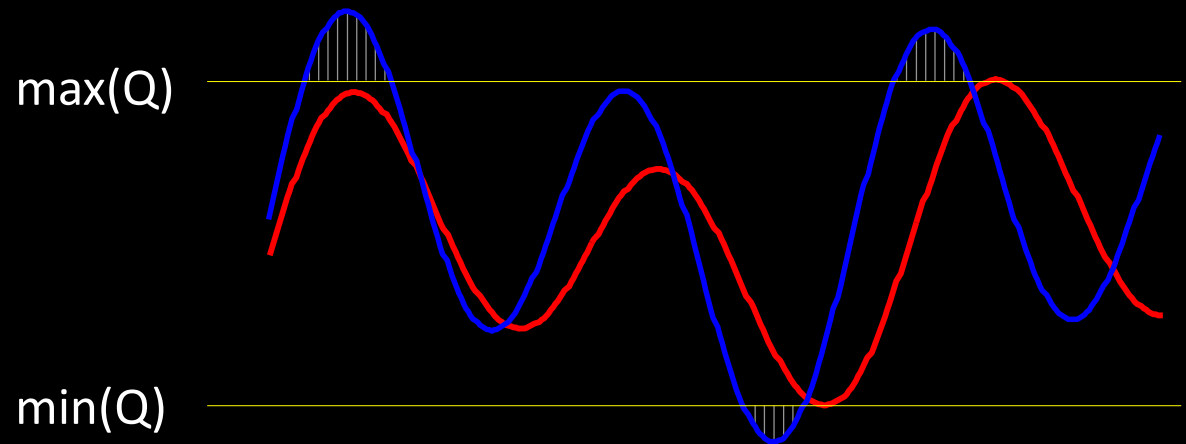
Lower Bound of Y_i



LB_Yi

$O(n)$ time

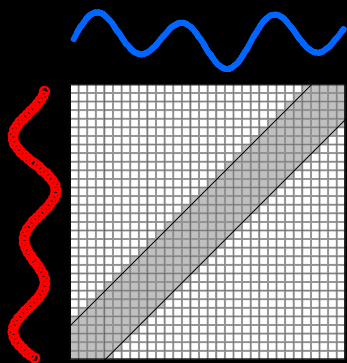
Yi, B, Jagadish, H & Faloutsos, C.
*Efficient retrieval of similar time
sequences under time warping.*
ICDE 98, pp 23-27.



The sum of the squared length of white lines represent the minimum contribution of the observations above and below the **yellow** lines.

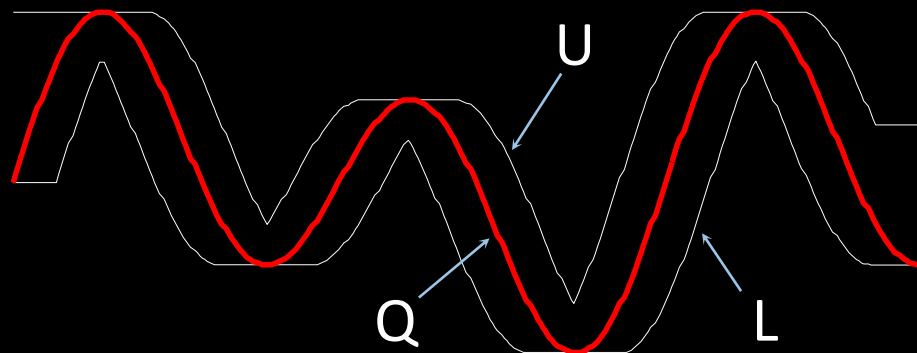


Lower Bound of Keogh



Sakoe-Chiba Band

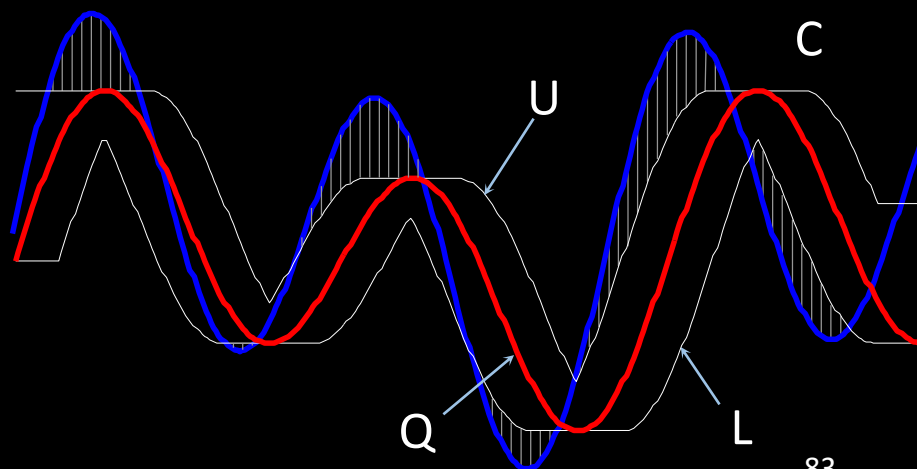
$$U_i = \max(q_{i-w} : q_{i+w})$$
$$L_i = \min(q_{i-w} : q_{i+w})$$



$O(n)$ time

Envelope-Based
Lower Bound

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

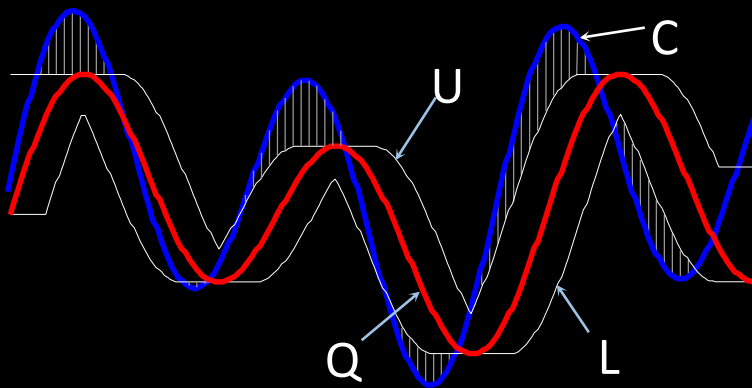


Reversing the Query/Data Role in LB_Keogh

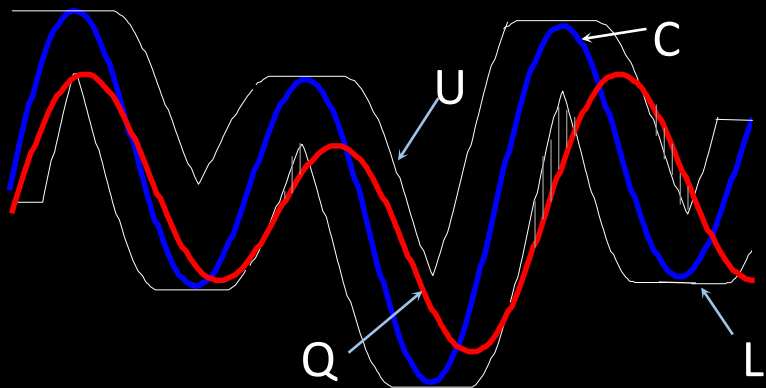
- Make LB_Keogh tighter
- Much cheaper than DTW
- U/L envelopes on the candidates can be calculated online or pre-calculated

$$\max(\text{LB_Keogh}^{EQ}, \text{LB_Keogh}^{EC})$$

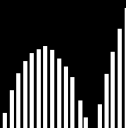
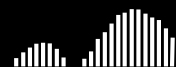
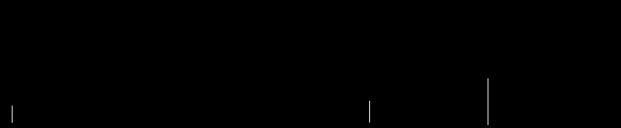
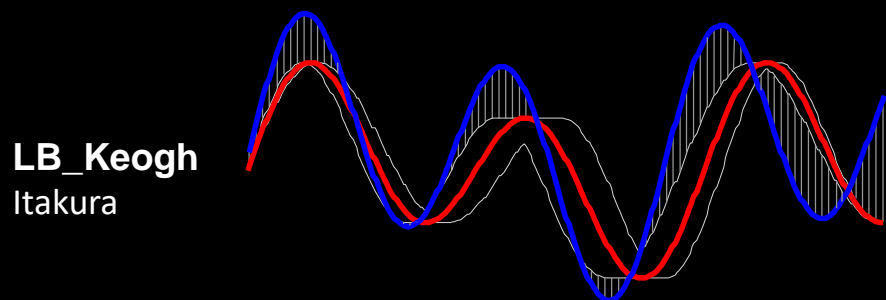
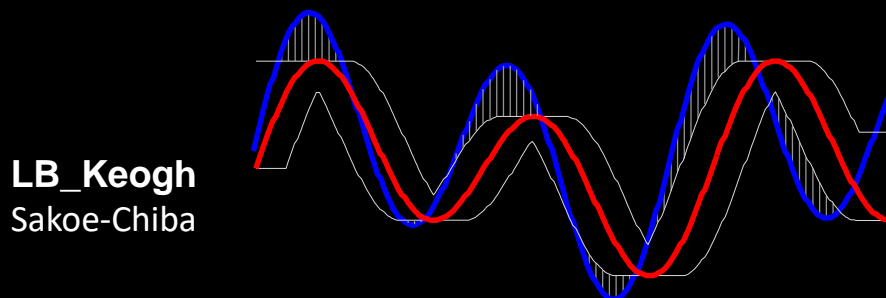
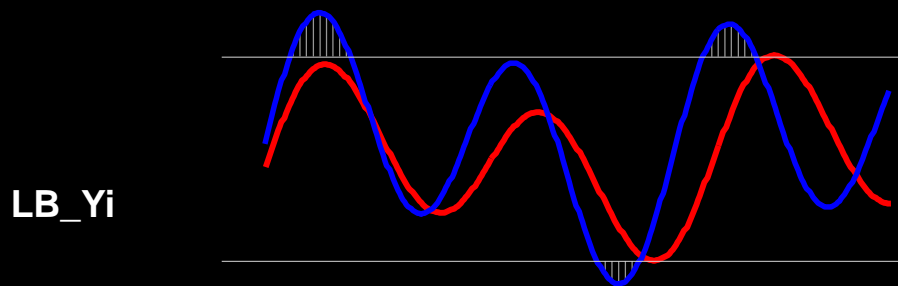
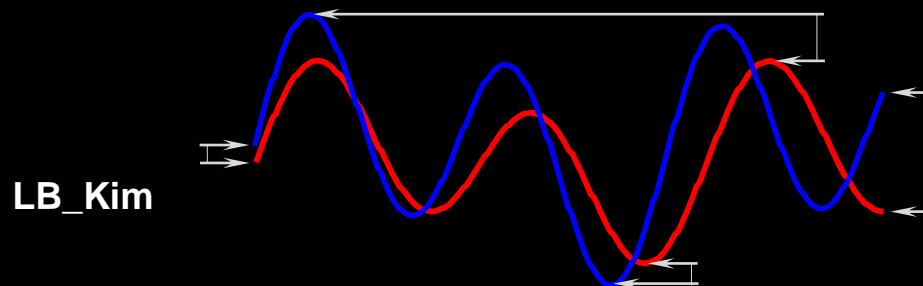
Envelop on Q



Envelop on C

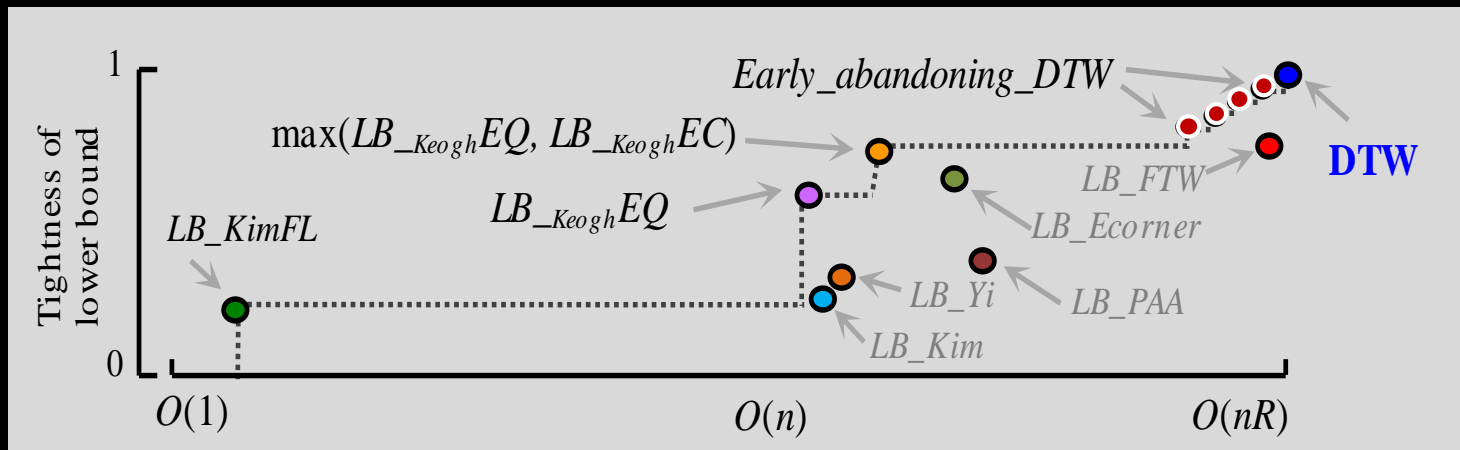


The tightness of the lower bound for each technique is proportional to the length of lines used in the illustrations



Cascading Lower Bounds

- At least 18 lower bounds of DTW was proposed.
- Use lower bounds only on the *Skyline*.
- Use the bounds on the skyline in cascade from least expensive to most expensive
- When unable to prune, use early abandoning techniques



99.9% of the time DTW is not calculated

Early Abandoning Techniques

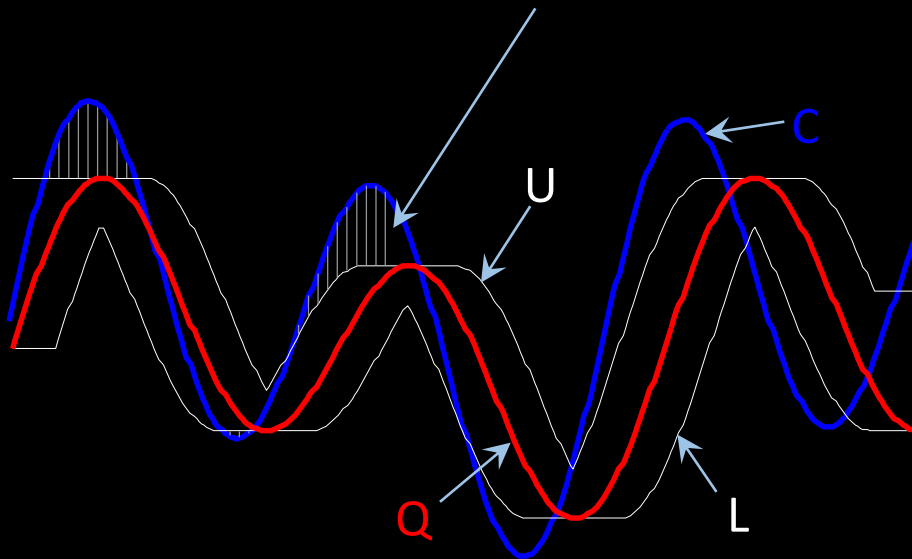
Abandon accumulating errors as soon as the current total is larger than the *best_so_far*

Four techniques to abandon early

1. Early Abandoning of LB_Keogh
2. Early Abandoning of DTW
3. Earlier Early Abandoning of DTW using LB_Keogh
4. Reordering Early Abandoning

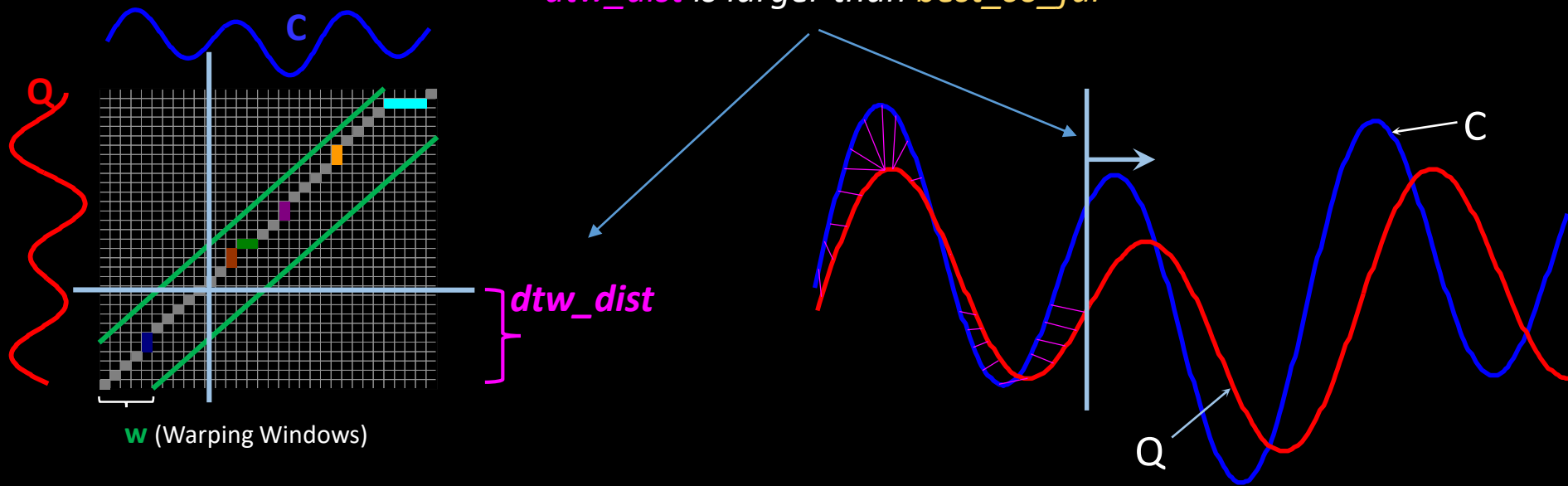
Early Abandoning of LB_Keogh

*Abandon the computation, when the accumulated error is larger than **best_so_far***



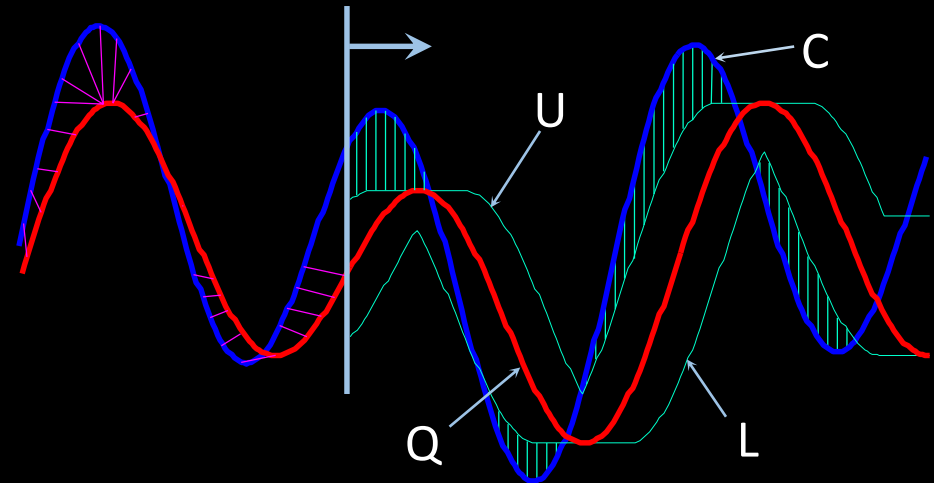
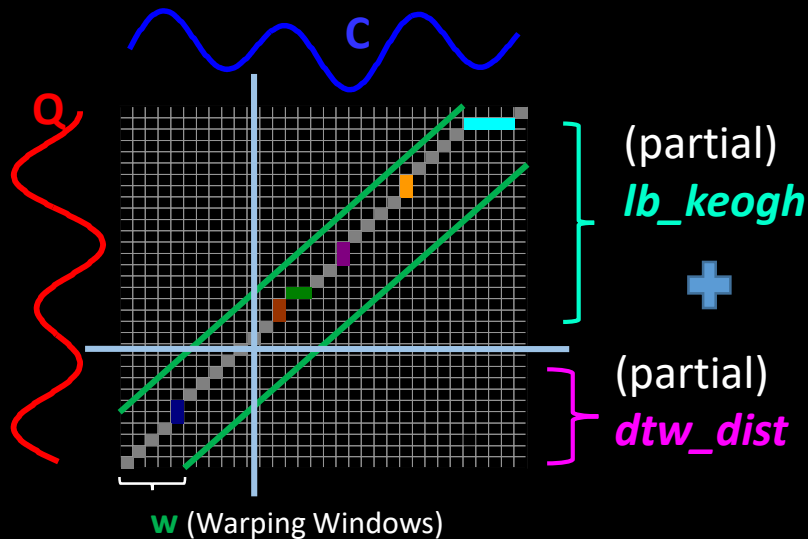
U, L are upper and lower envelopes of Q

Early Abandoning of DTW



Earlier Early Abandoning of DTW using LB_Keogh

Abandon the computation, when the
 $dtw_dist + lb_keogh$ is larger than $best_so_far$

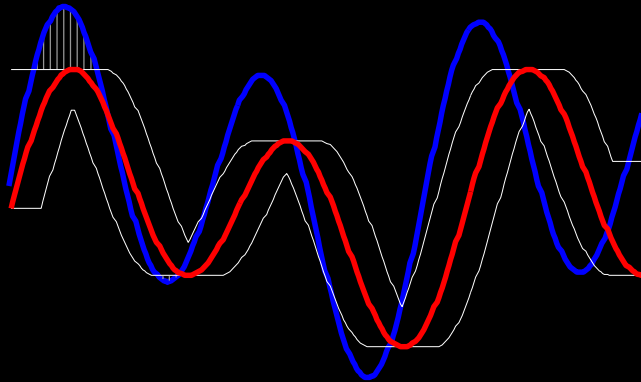


U, L are upper and lower envelopes of Q

Reordering Early Abandoning

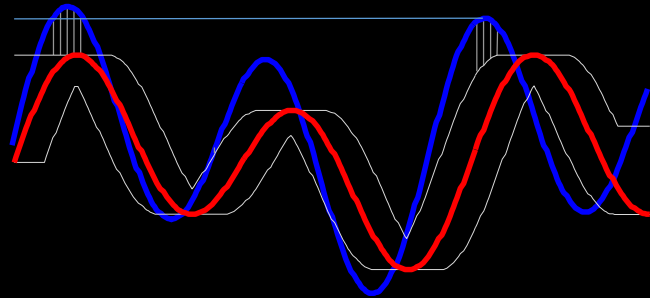
- We don't have to compute LB from left to right.
- Order points by expected contribution.

1 2 3 4 5 6 7 8 9

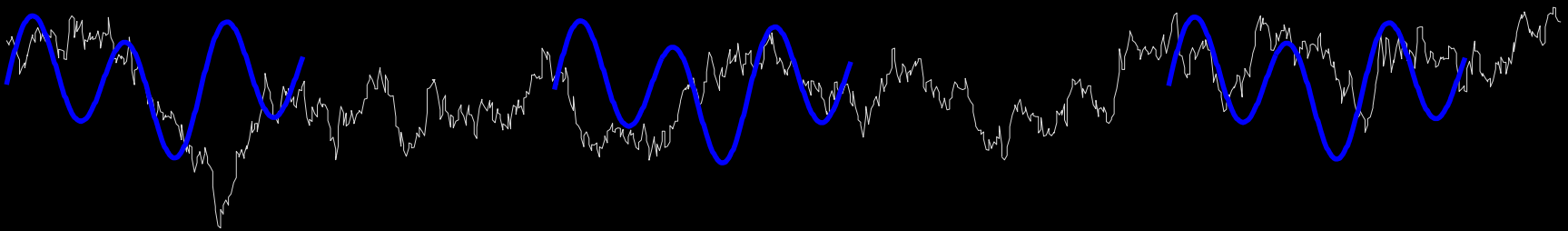


7 3 1 2 6

8 4 5 9



Idea



- Order by the absolute height of the query point.

Summary of the techniques

Group-1 Techniques

- Early Abandoning of LB_Keogh
- Early Abandoning of DTW
- Ear^{lier} Early Abandoning of DTW using LB_Keogh



Group-2 Techniques

- Just-in-time Z-normalizations
- Reordering Early Abandoning
- Reversing LB_Keogh
- Cascading Lower Bounds



UCR Suite

Code and data is available at:

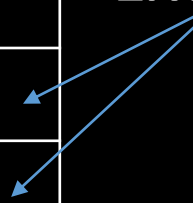
www.cs.ucr.edu/~eamonn/UCRsuite.html

Experimental Result: Random Walk

- Random Walk: Varying size of the data, $|Q| = 128$

	Million (Seconds)	Billion (Minutes)	Trillion (Hours)
DTW-Naive	75.21	<u>1,252.2</u>	<u>20,869</u>
Group-1	2.447	38.14	<u>472.80</u>
Group-1 and 2	0.159	1.83	34.09

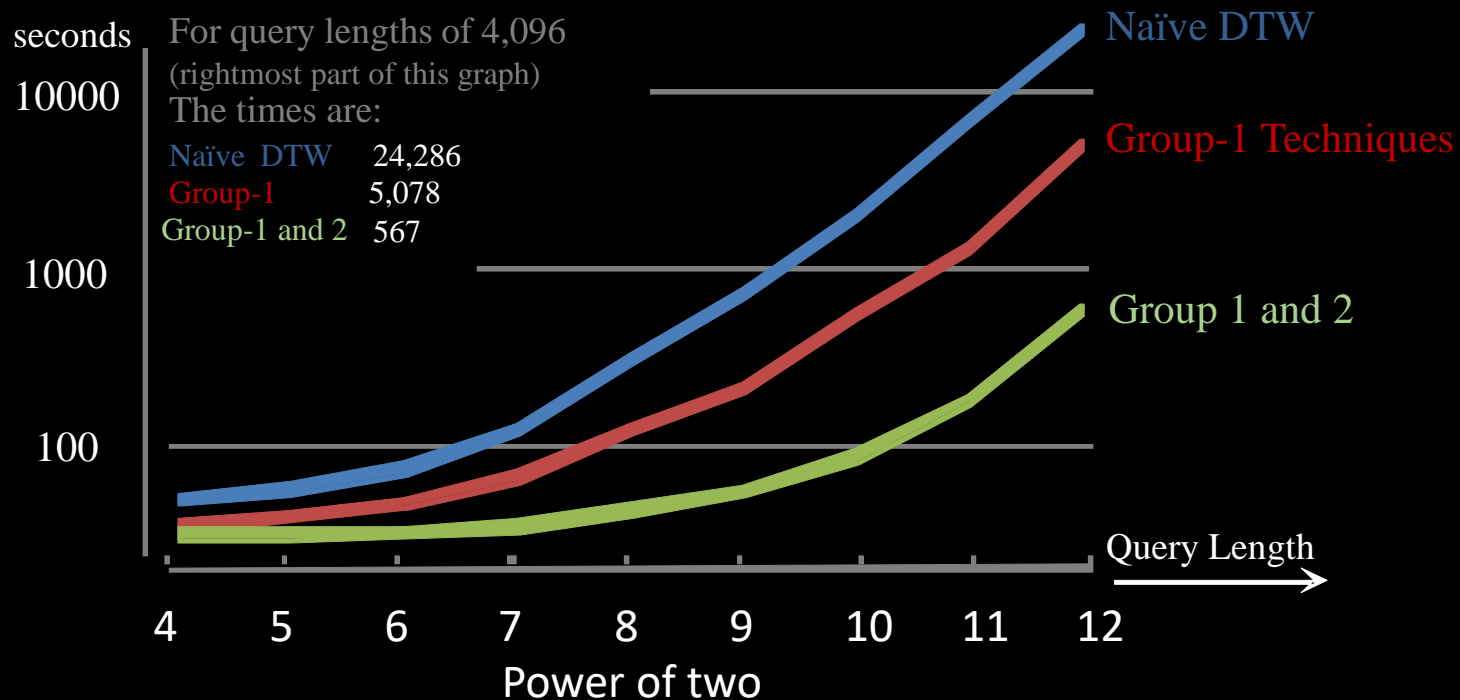
Extrapolated



- Experiments performed on a commodity machine
- Disk is accessed sequentially as the algorithm is invariant to data order
- The computation is performed while the next disk block is read

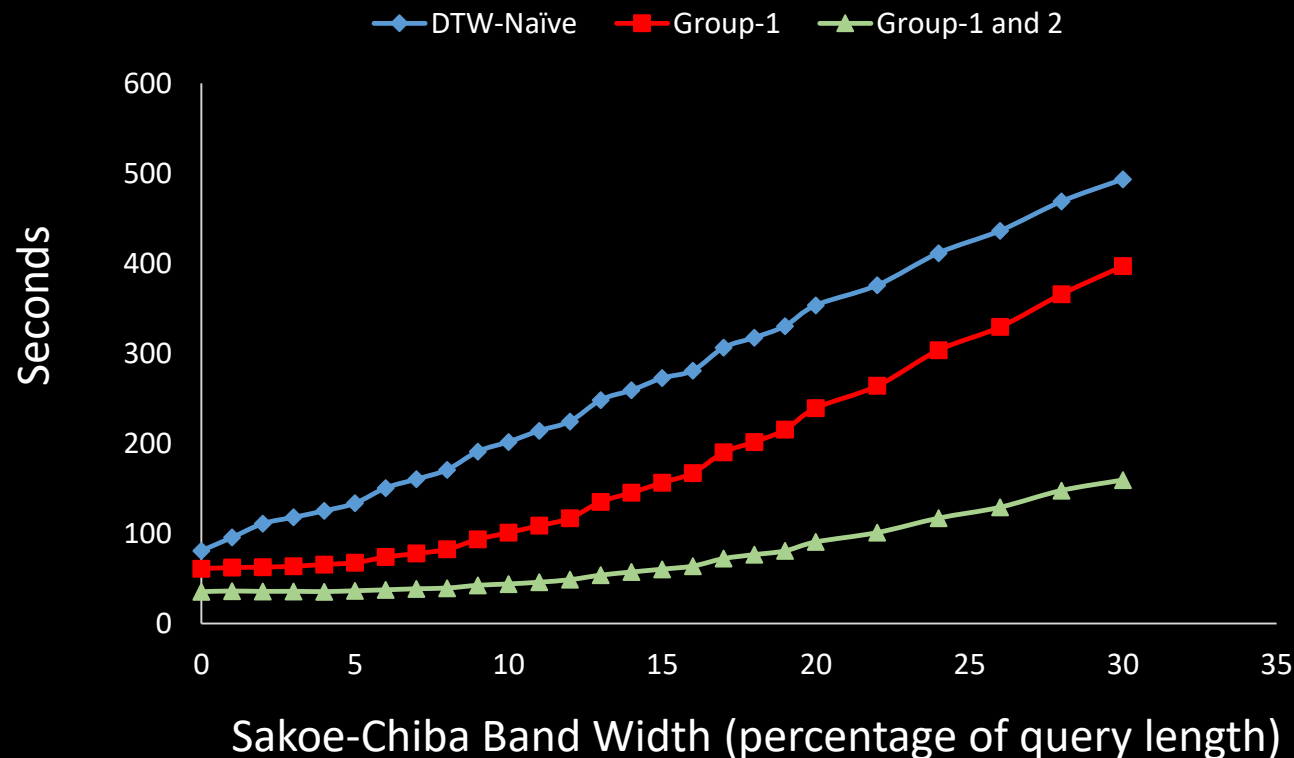
Experimental Result: Random Walk

- Random Walk: Varying size of the query, $n = 20$ million



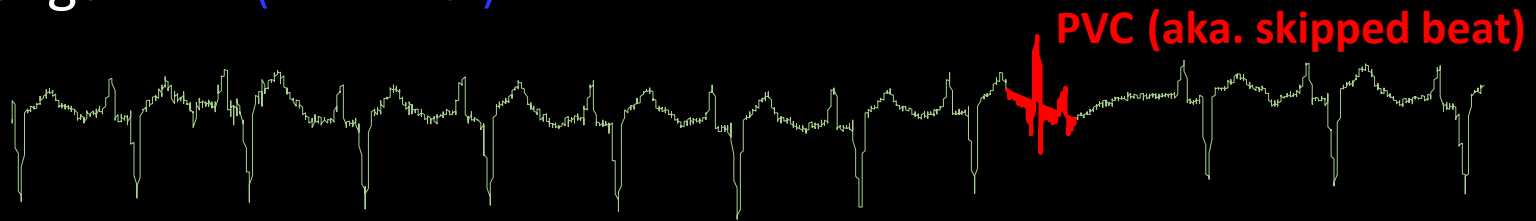
Experimental Result: Random Walk

- Random Walk: Varying size of the band



Experimental Result: ECG

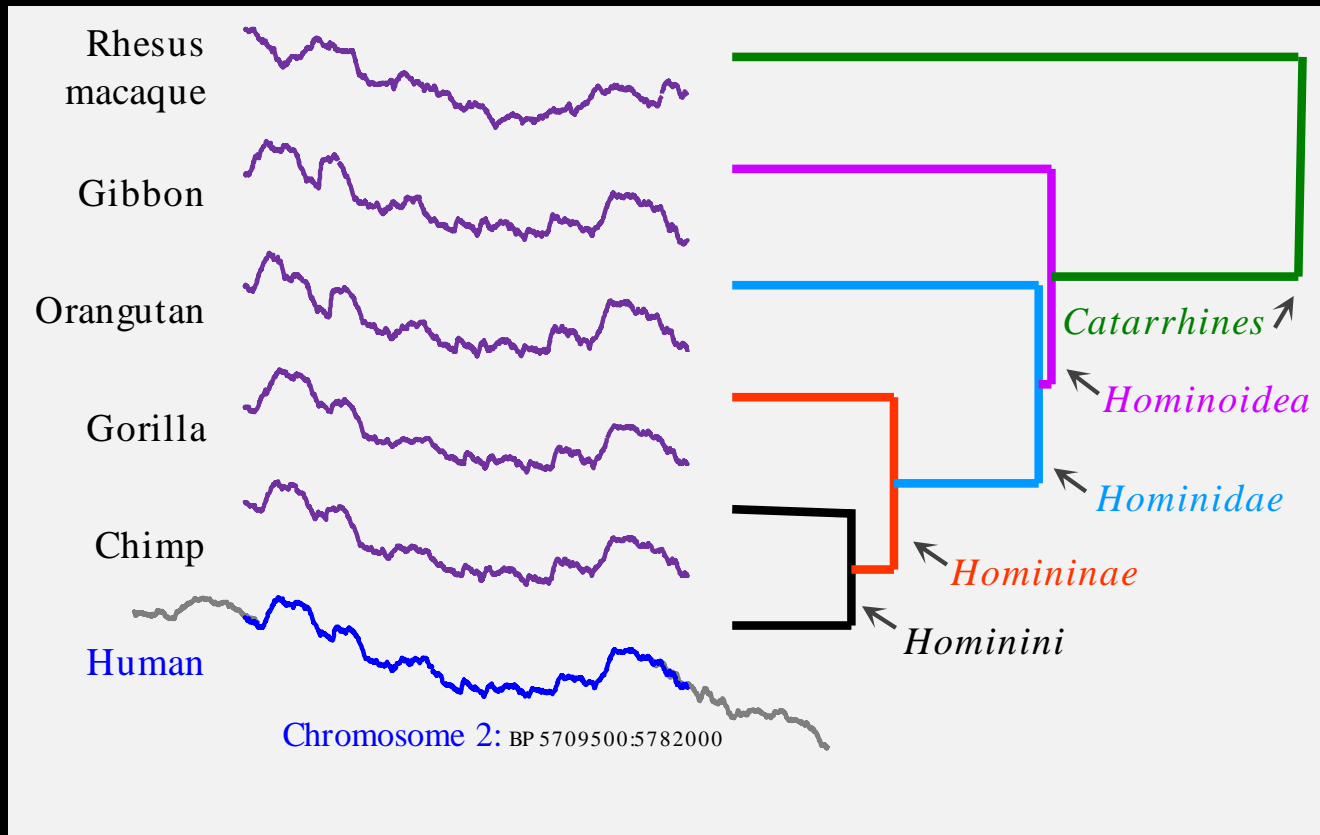
- Data: **One year** of Electrocardiograms 8.5 billion data points.
- Query: Idealized Premature Ventricular Contraction (PVC) of length 421 ($w=21=5\%$).



	Group-1	Group 1 & 2
ECG	49.2 hours	18.0 minutes

Experimental Result: DNA

- Query: Human Chromosome 2 of length 72,500 bps
- Data: Chimp Genome 2.9 billion bps
- Time: UCR Suite 14.6 hours

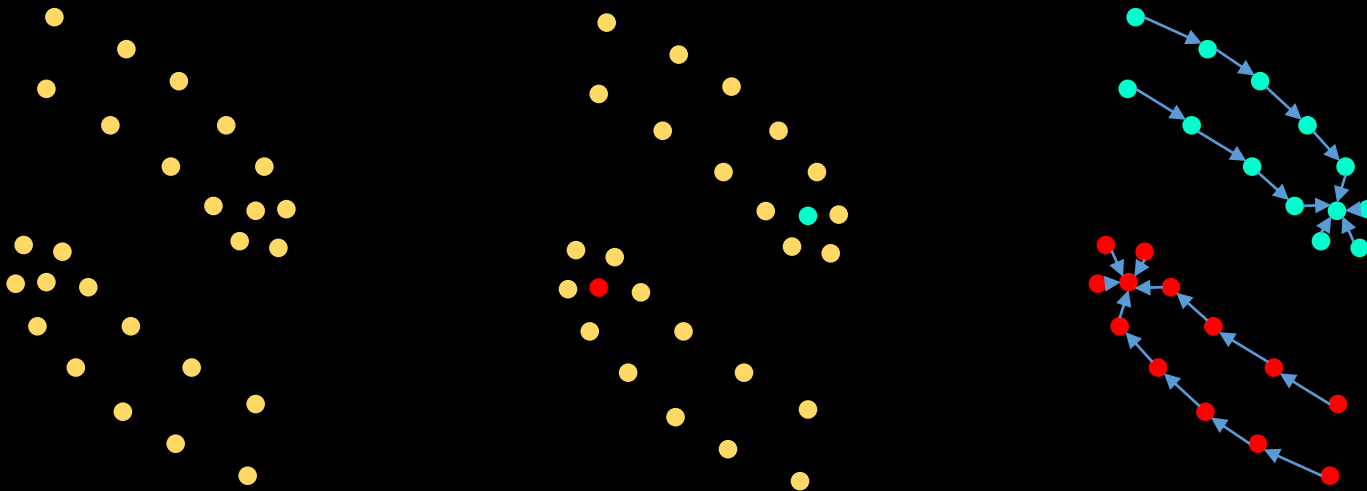


What can be made fast?

- One-to-One comparison
 - Exact Implementation and Constraints
 - Efficient Approximation
 - Exploiting Sparsity
- **One-to-Many comparisons**
 - Nearest Neighbor Search
 - In subsequences of a long time series
 - Density Estimation
 - In clustering
 - Averaging Under Warping
 - In classification
- Many-to-Many comparisons
 - All-pair Distance Calculations

Density based clustering

- Density Peaks (DP)* Algorithm
 - Find the densities of every point to pick cluster centers
 - Connect every point to the nearest higher density point



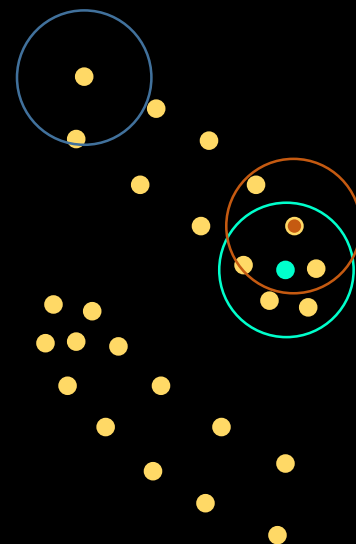
*Rodriguez, A., & Laio, A. (2014). *Clustering by fast search and find of density peaks*. Science, 344(6191), 1492-1496.

Range Search/Density Estimation

- Density is estimated by the number of points within a radius/threshold t

Algorithm Bounding_Range_Search(Q, t)

```
1. for all sequences  $C_i$  in database
2.   LB_dist = lower_bound_distance ( $C_i, Q$ )
3.   if LB_dist <  $t$ 
4.     UB_dist = upper_bound_distance ( $C_i, Q$ )
5.     if UB_dist <  $t$  then output  $C_i$ 
6.   else
7.     true_dist = DTW( $C_i, Q$ )
8.     if true_dist <  $t$ 
9.       output  $C_i$ 
10.    endif
11.  endif
12. endif
13. endfor
```



Try to use an upper bound to identify a point within the range

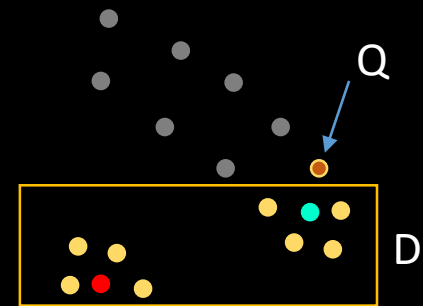


Density Connectedness

- Distance between a pair of points is an upper bound of the NN distance from both of the points

Algorithm Bounding_Scan(D,Q)

```
1. best_so_far = min(upper_bound_NN_distance(D,Q))
2. for all sequences in D
3.   LB_dist = lower_bound_distance(  $C_i$ , Q);
4.   if LB_dist < best_so_far
5.     true_dist = DTW(  $C_i$ , 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
```

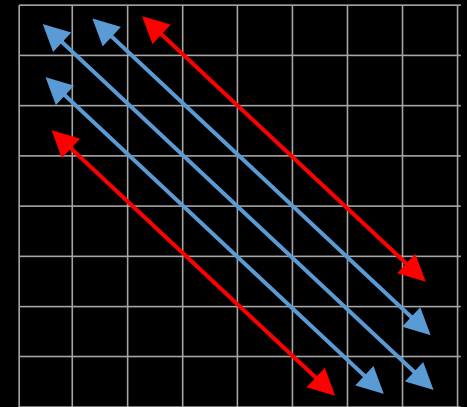
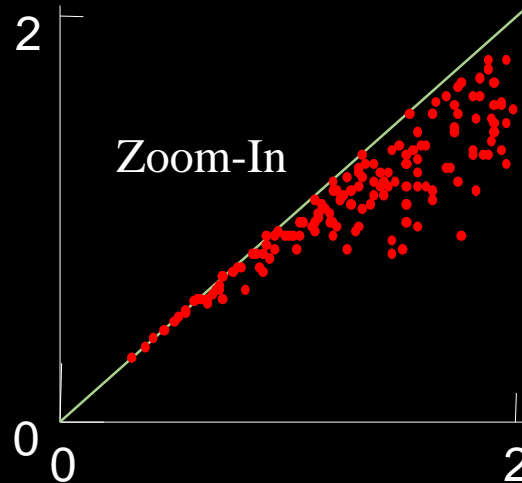
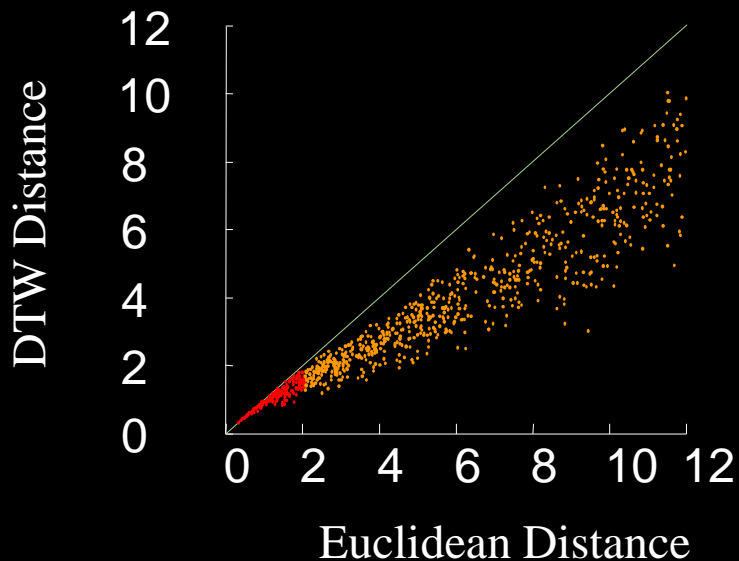


Try to use an upper bound to the NN distance as the *best_so_far*



Upper bounding

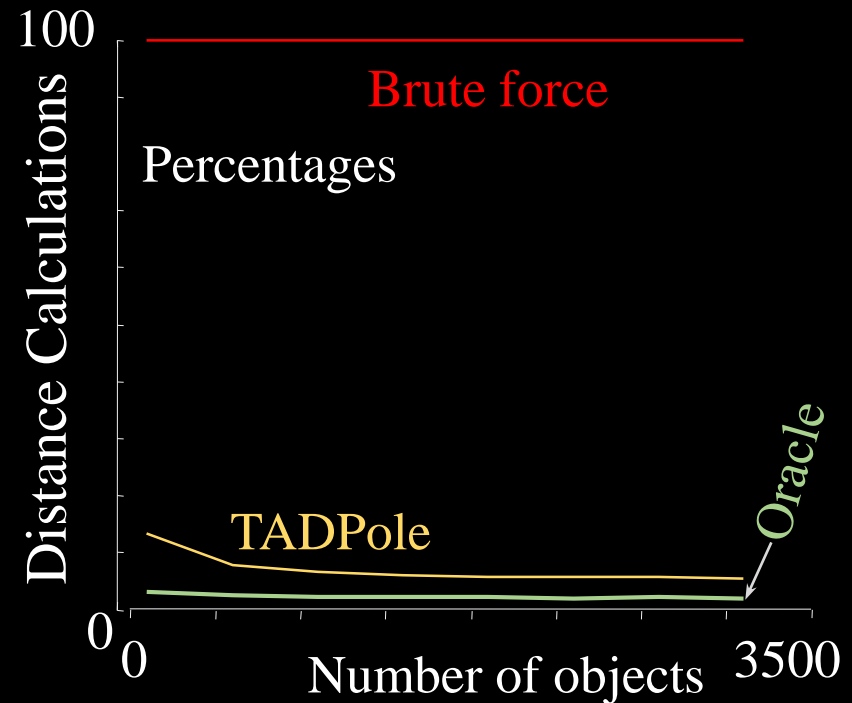
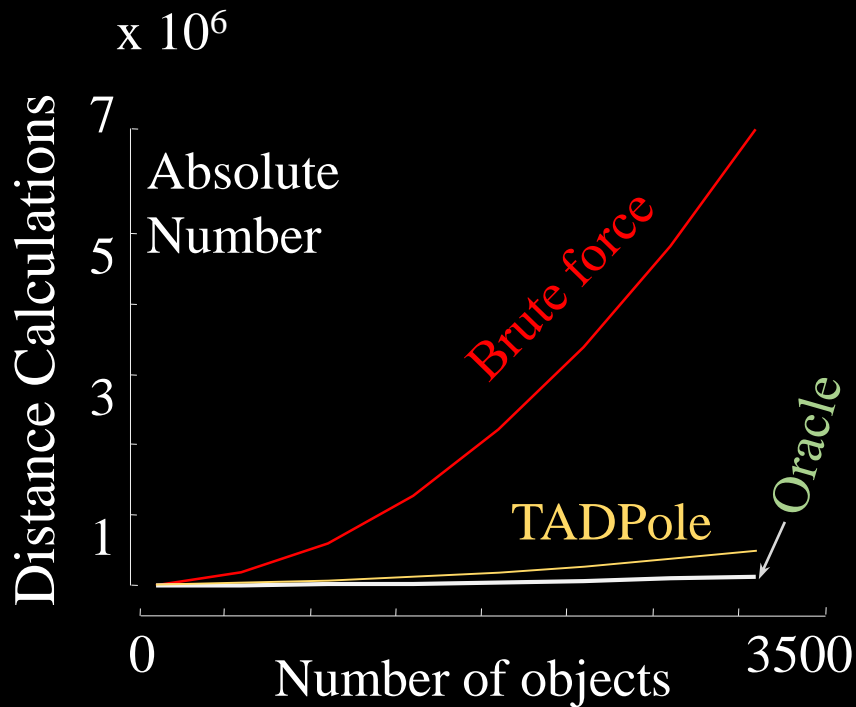
- Euclidean distance is a trivial upper bound
- DTW distance in a band w is an upper bound for DTW distance in band $w+1$



Speedup by upper bounds

Density Peak: 9 Hours

TADPole: 9 minutes



StarLightCurves dataset

What can be made fast?

- One-to-One comparison
 - Exact Implementation and Constraints
 - Efficient Approximation
 - Exploiting Sparsity
- **One-to-Many comparisons**
 - Nearest Neighbor Search
 - In a database of independent time series
 - In subsequences of a long time series
 - Density Estimation
 - In clustering
 - Averaging Under Warping
 - In classification
- Many-to-Many comparisons
 - All-pair Distance Calculations

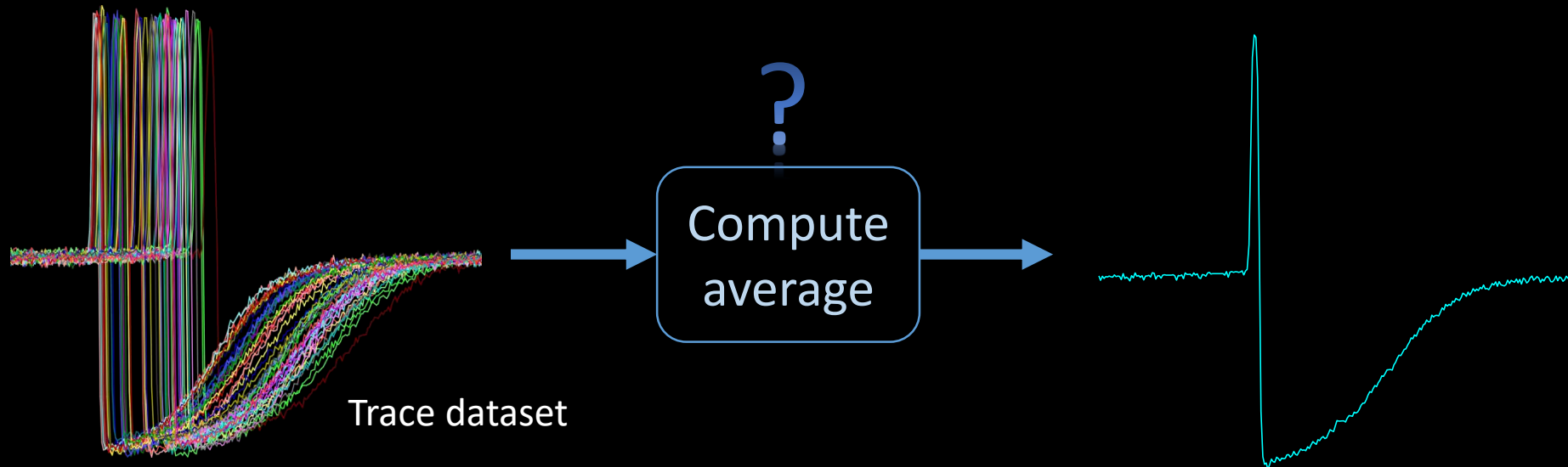
Data Reduction for 1NN Classification

- The training set is reduced to a smaller set keeping a representative set of labeled instances
- Smaller training set entails performance gain
- Smaller training set may gain accuracy if noisy instances are filtered effectively
- **Reduction methods**
 - Random Selection
 - Rank the instances and take top-K
 - Cluster instances based on proximity and take representative from each cluster

Many clustering algorithms require finding a centroid of two or more instances

The issue is then:

➤ *How to average time series consistently with DTW?*



Mathematically, the mean \bar{o} of a set of objects O embedded in a space induced by a distance d is:

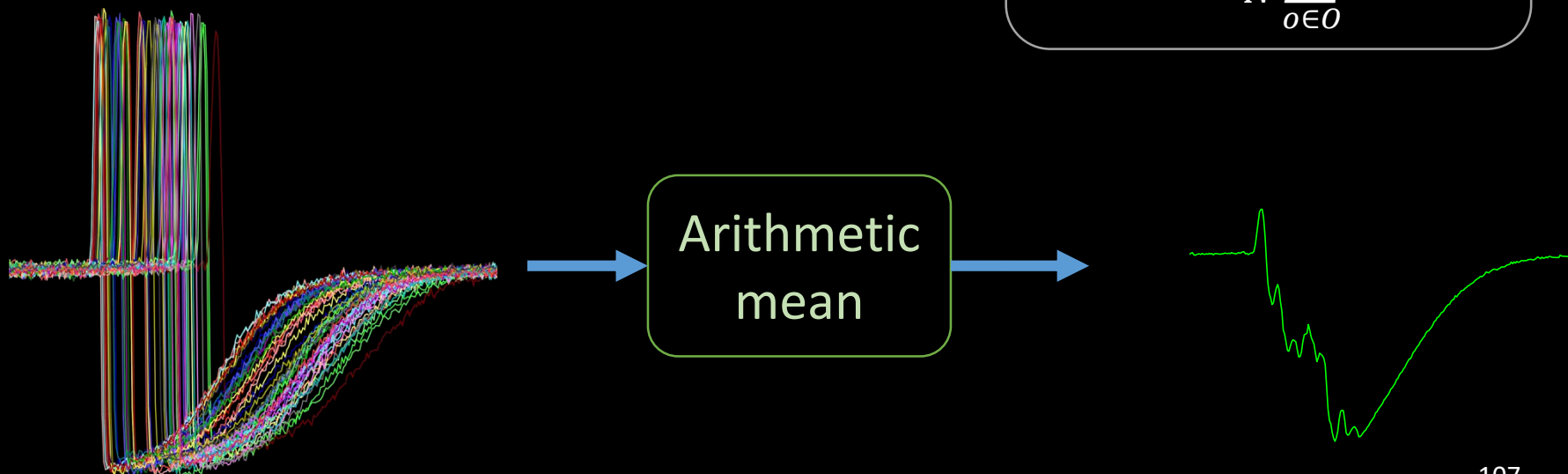
$$\arg \min_{\bar{o}} \sum_{o \in O} d^2(\bar{o}, o)$$

The mean of a set minimizes the sum of the squared distances

If d is the Euclidean distance

The *arithmetic mean* solves the problem exactly

$$\bar{o} = \frac{1}{N} \sum_{o \in O} o$$



To solve the optimization problem for DTW distance, we need to perform simultaneous alignment of many time series.

But, finding the optimal multiple alignment:

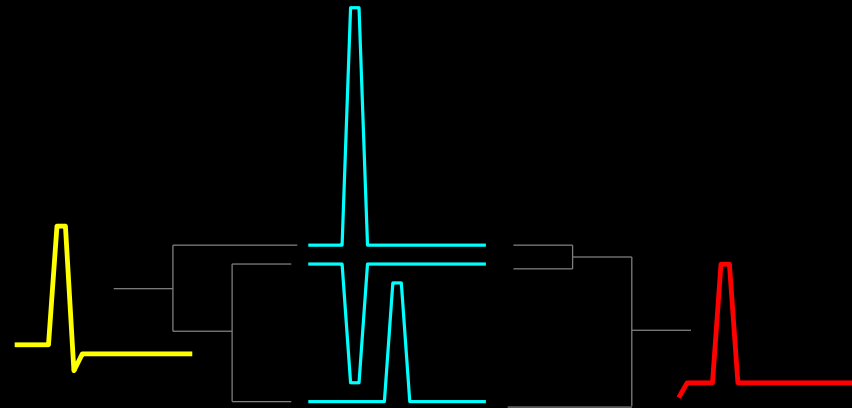
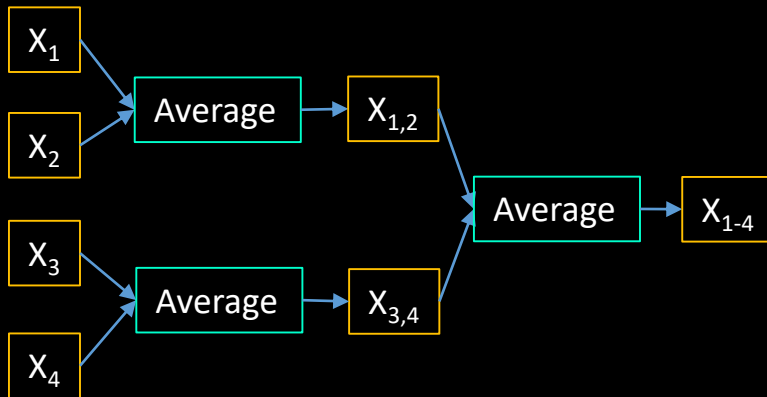
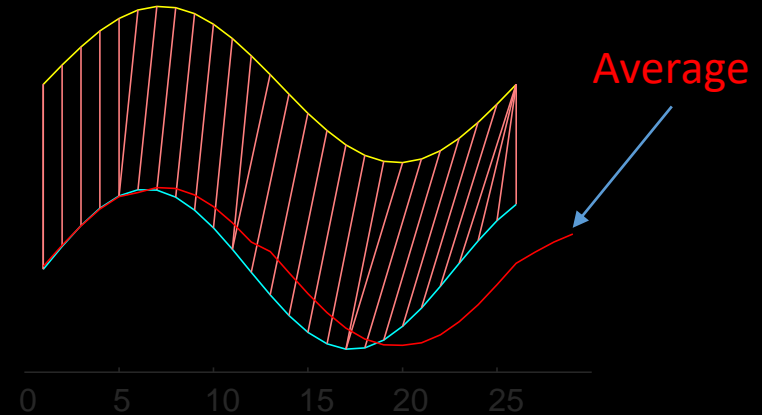
1. Is **NP-complete** [a]
2. Requires $O(L^N)$ operations
 - L is the length of the sequences (≈ 100)
 - N is the number of sequences ($\approx 1,000$)

⇒ Efficient solutions will be heuristic

- Pairwise Averaging
- DTW Barycenter Averaging (DBA)

Pairwise averaging for DTW

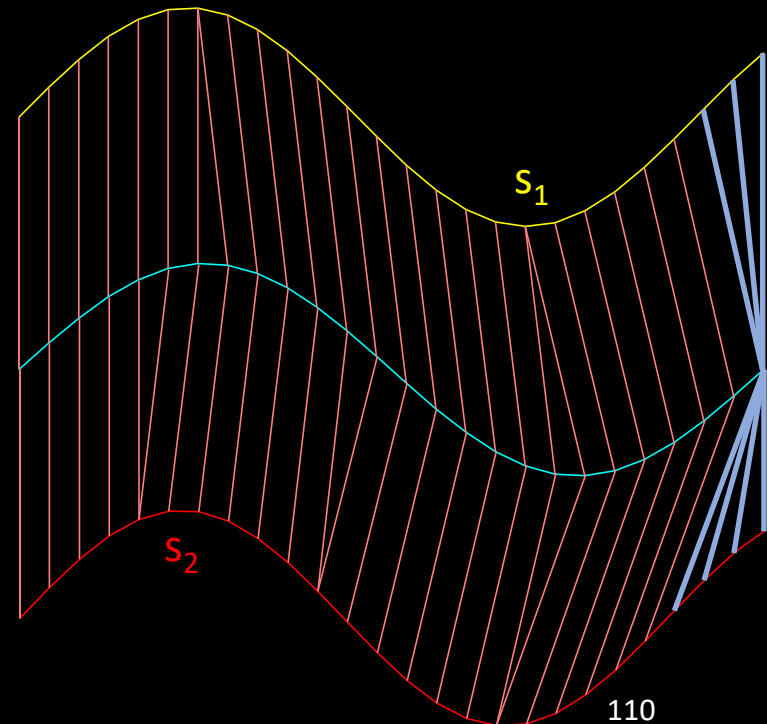
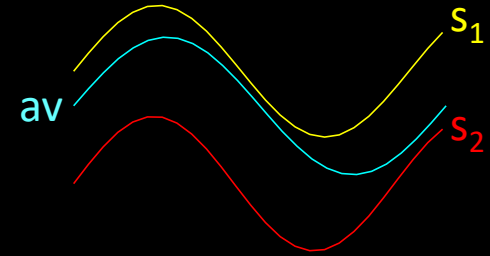
- Average each alignment between the two time series
- Commonly increases the length
- Chaining can produce average over a set
- The operation is not associative, the average produced depends on the order



DTW Barycenter Averaging (DBA)

Algorithm DBA(D,av)	
1	Iterate until convergence
2	for each sequence s_i in D
3	$A_i = \text{GetAlignment}(\text{DTW}(s_i, \text{av}))$
4	for each observation j in av
5	$\text{av}[j] = \text{mean}([A_1[j] \ A_2[j] \ A_3[j] \ \dots \ A_n[j]])$

- Does not increase length
- Does not depend on the order of the points



Experimental Evaluation on Insect Data

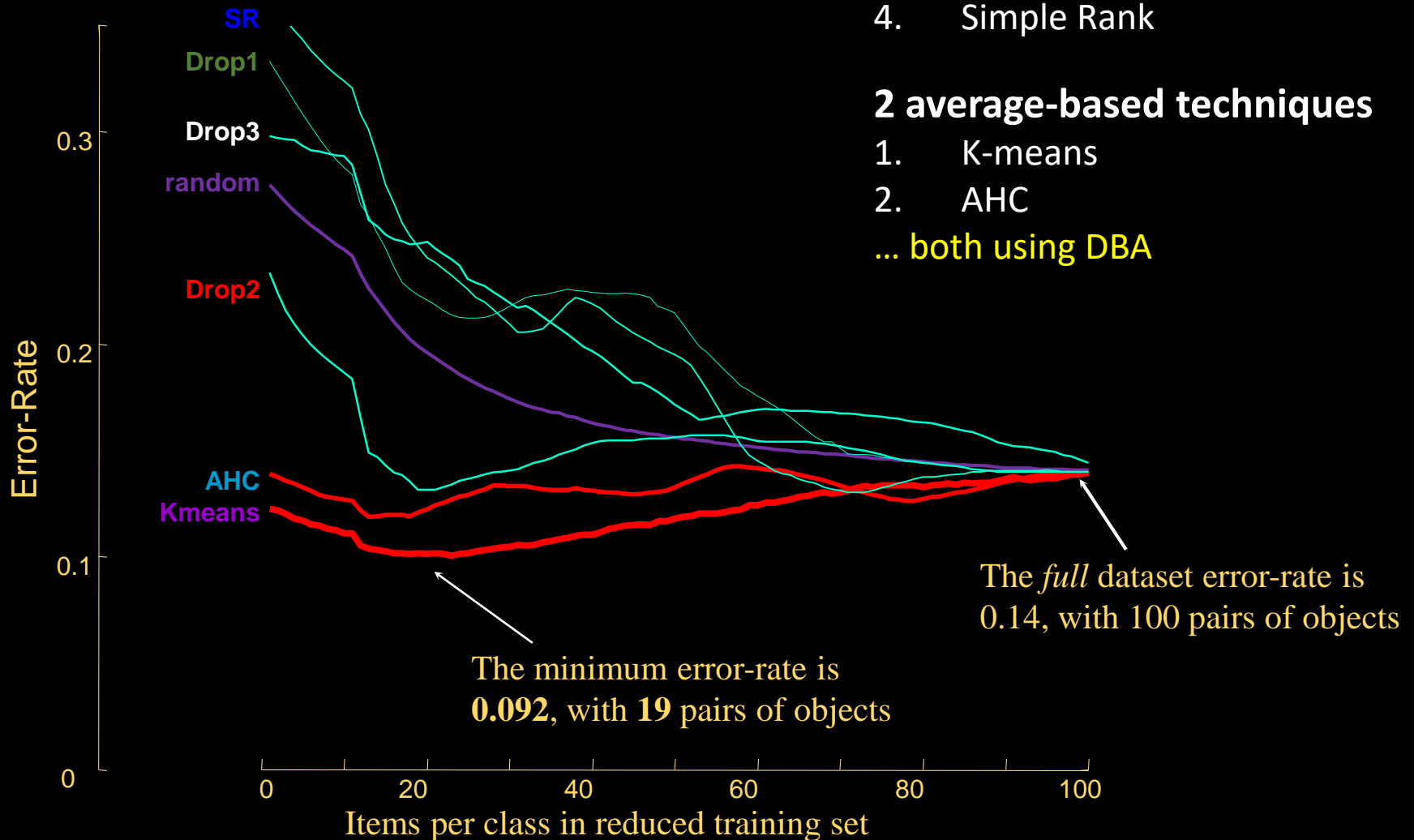
4 rank-based competitors

1. Drop 1
2. Drop 2
3. Drop 3
4. Simple Rank

2 average-based techniques

1. K-means
2. AHC

... both using DBA



What can be made fast?

- One-to-One comparison
 - Exact Implementation and Constraints
 - Efficient Approximation
 - Exploiting Sparsity
- One-to-Many comparisons
 - Nearest Neighbor Search
 - In a database of independent time series
 - In subsequences of a long time series
 - Density Estimation
 - In clustering
 - Averaging Under Warping
 - In classification
- Many-to-Many comparisons
 - All-pair Distance Calculations

Speeding up DTW: many-to-many

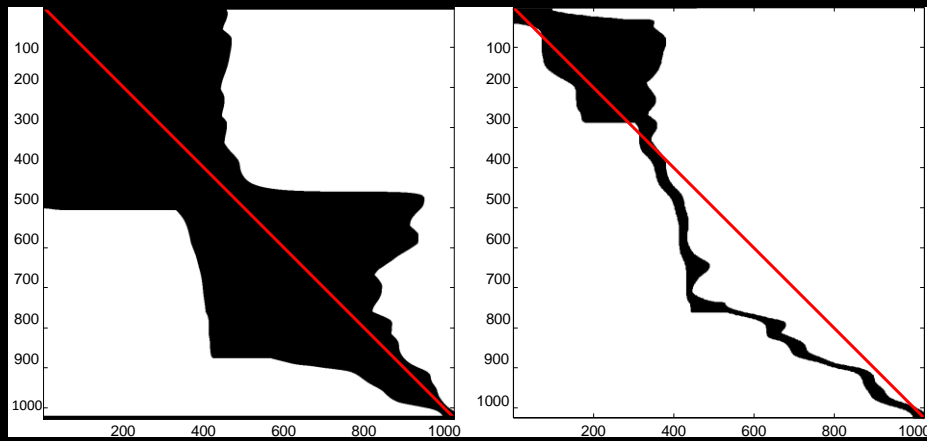
- Several variants
 - Self-join within a threshold - top-K Self-join
 - Use similarity search techniques as subroutine
 - Application: Motif discovery [a], Discord discovery
 - A/B Join within a threshold - top-K A/B Join
 - Use similarity search techniques as subroutine
 - Application: Motion Stitching [b]
 - All-pair distance matrix
 - Use techniques to speedup one-to-one comparisons
 - Application: Hierarchical Clustering

[a] N Chavoshi, H Hamooni, A Mueen, "DeBot: Real-Time Bot Detection via Activity Correlation" UNM Technical Report

[b] Y. Chen, G. Chen, K. Chen and B. C. Ooi, "Efficient Processing of Warping Time Series Join of Motion Capture Data," ICDE, 2009, pp. 1048-1059.

PrunedDTW: speeding up all-pair distance matrix calculation

- Two types of pruning when calculating DTW matrix
- Exact method



UB = Euclidean distance

UB = DTW distance

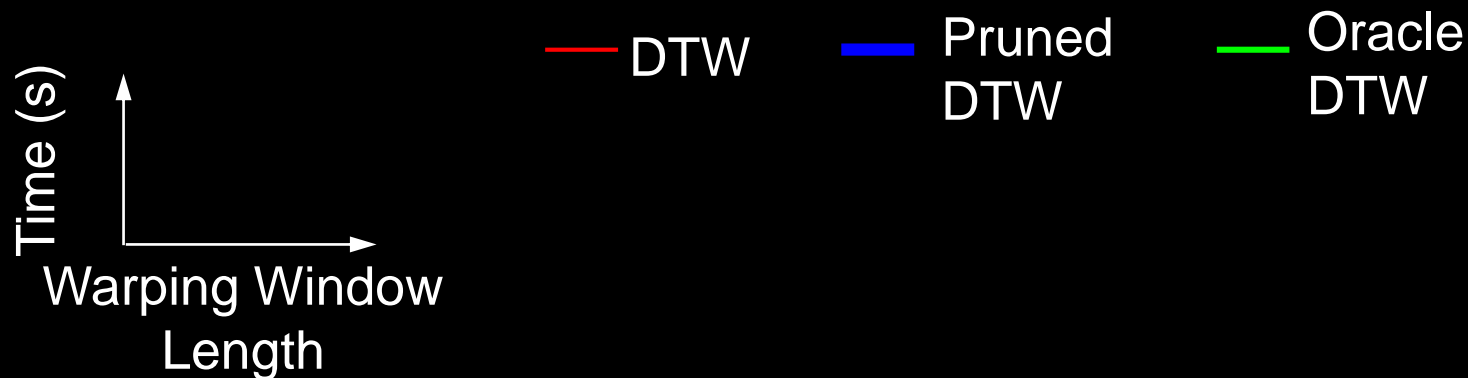
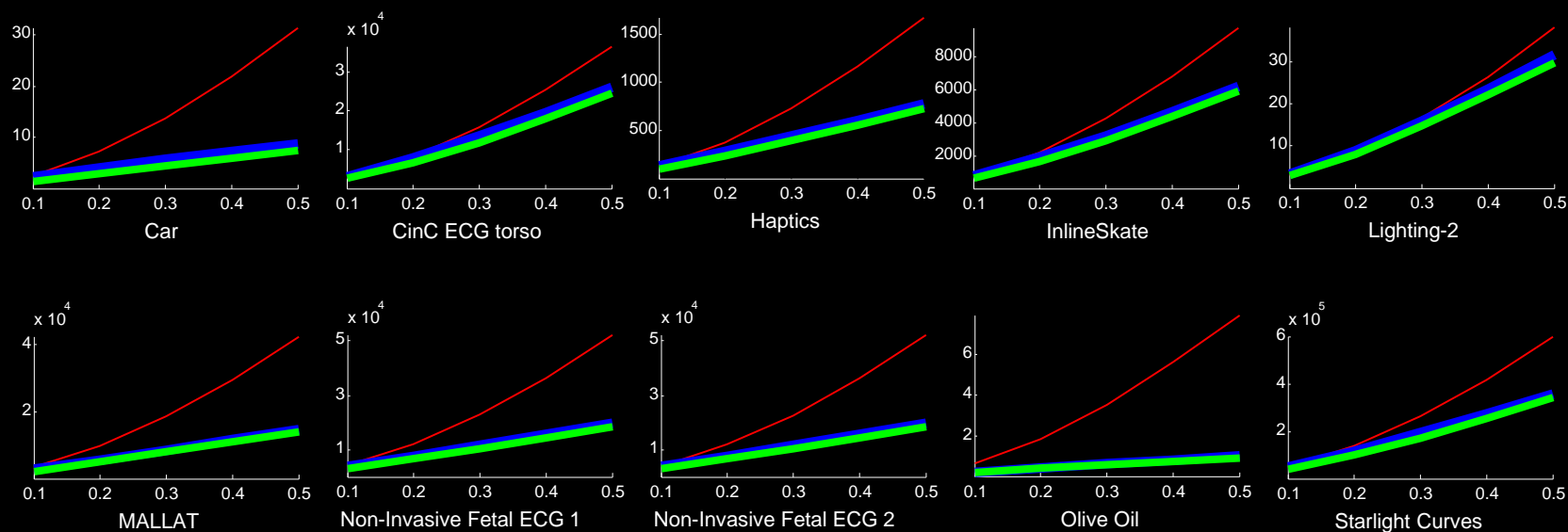
	0	1	2	3	4	5	6
3	∞			$\leq UB$			
4	∞	$> UB$	$> UB$	$\leq UB$	$\leq UB$		
5	∞					$\leq UB$	
6	∞						$\leq UB$

Lower triangle pruning

	0	1	2	3	4	5	6
0	0	∞	∞	∞	∞	∞	∞
1	∞	$\leq UB$	$\leq UB$	$\leq UB$	$> UB$	$> UB$	$> UB$
2	∞		$\leq UB$	$\leq UB$	$> UB$		
3	∞			$\leq UB$			

Upper triangle pruning

Experiments



Future of Similarity Search (1 of 2)

- Adapt to data domains
 - Incorporate scientific knowledge in similarity measures
 - Phase-locking
 - Linear attenuation
 - Improvise the search techniques to work with data domains
 - Apply constraints in lower bounding and early abandoning techniques: band-limited signals, minimum lag between signals, etc.
- Scalability via distributed computation
 - Preserve algorithmic improvements upon distribution. Do not settle with larger pool of workers, when smaller pool is enough

Future of Similarity Search (2 of 2)

- Streaming Analytics
 - Keep pace with the data rate: worst case is the average case
 - Reduce memory footprint and achieve interactivity
- Predictive Analytics
 - Similarity based predictions
 - Minimize lag before prediction

Additional References

- H. Sakoe and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition,” IEEE Trans. Acoust. Speech, Lang. Process., vol. 26, no. 1, pp. 43–50, 1978.
- Mustafa S. Çetin, Abdullah Mueen, Vince D. Calhoun: Shapelet Ensemble for Multi-dimensional Time Series. SDM 2015: 307-315
- X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, “Experimental comparison of representation methods and distance measures for time series data,” Data Min. Knowl. Discov., vol. 26, no. 2, pp. 275–309, 2013.
- D. Sart, A. Mueen, W. Najjar, V. Niennattrakul, and E. Keogh, “Accelerating Dynamic Time Warping Subsequence Search with GPUs and FPGAs. ICDM 2010,” in Proceedings - IEEE International Conference on Data Mining, ICDM, 2010, pp. 1001–1006.
- Abdullah Mueen, Hossein Hamooni, Trilce Estrada: Time Series Join on Subsequence Correlation. ICDM 2014: 450-459
- Ira Assent, Marc Wichterich, Ralph Krieger, Hardy Kremer, and Thomas Seidl. 2009. Anticipatory DTW for efficient similarity search in time series databases. Proc. VLDB Endow. 2, 1 (August 2009), 826-837.

Conclusion

- Time series similarity search is no longer a slow operation to build data mining algorithms on
- Lockstep and Elastic measures both have applications in various domains. Their usages are limited only by our imagination.

