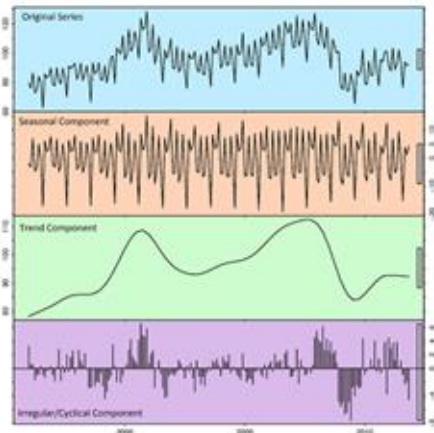


Time Series - Similarity, Distances, Transformations and Clustering

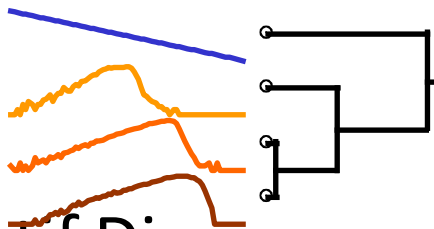


What We Can Do With Time Series?

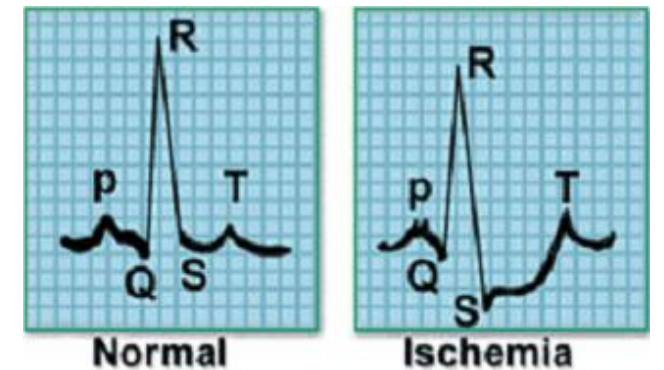
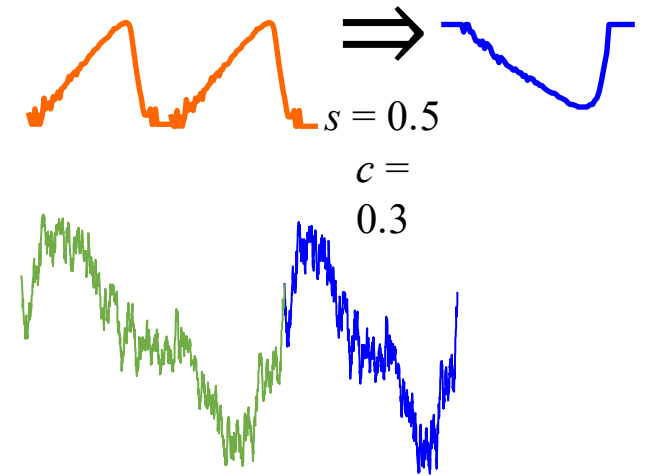
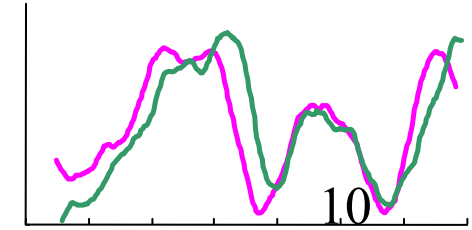
- Trends, Seasonality



- Clustering

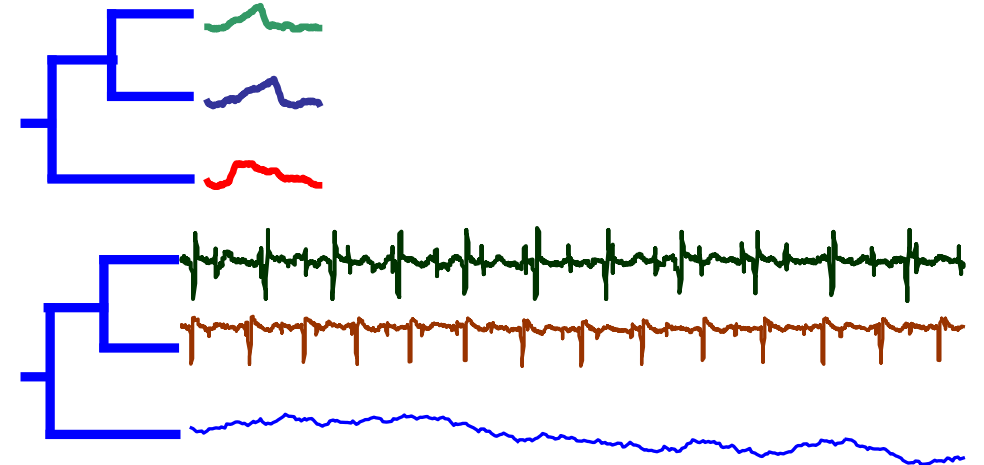


- Motif Discovery



Similarity

- All these problems require similarity matching.
- What is Similarity?
 - It is the quality or state of being similar, likeness, resemblance, as a similarity of features.
- In time series analysis we recognize two kinds of similarity:
 - Similarity at the level of shape
 - Similarity at the structural level



Shape-based Similarities

Euclidean Distance

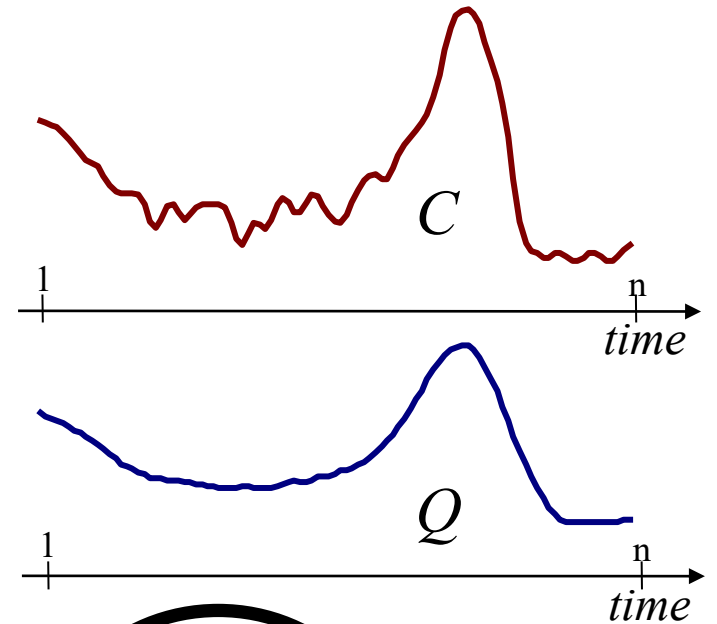
- Given two time series:

- $Q = q_1 \dots q_n$
- $C = c_1 \dots c_n$

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

- $T1 = \langle 56, 176, 110, 95 \rangle$
- $T2 = \langle 36, 126, 180, 80 \rangle$

$$D(T1, T2) = \text{sqrt} [(56-36)^2 + (176-126)^2 + (110-180)^2 + (95-80)^2]$$

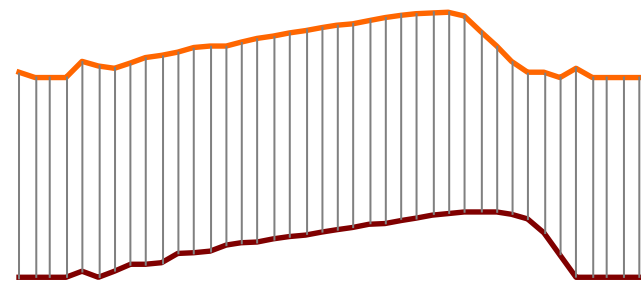


Problems with Euclidean Distance

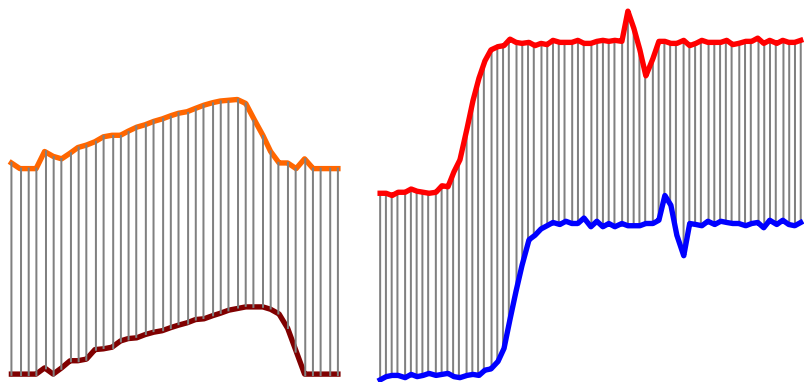
- Euclidean distance is very sensitive to “distortions” in the data.
- These distortions are dangerous and should be removed.
- Most common distortions:
 - Offset Translation
 - Amplitude Scaling
 - Linear Trend
 - Noise
- They can be removed by using the appropriate transformations.

Dynamic Time Warping

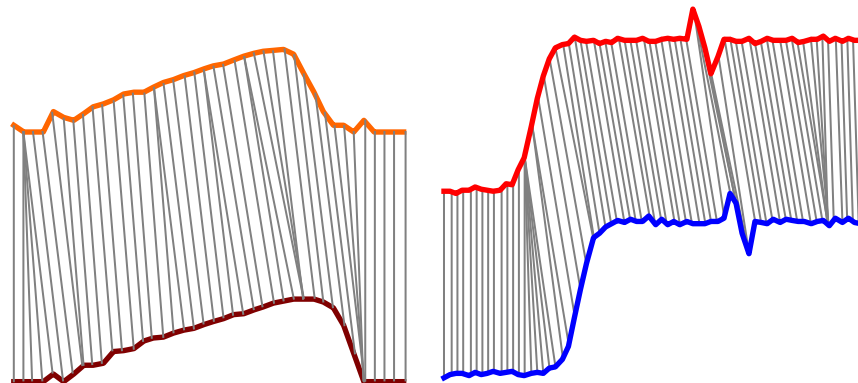
- Sometimes two time series that are conceptually equivalent evolve at different speeds, at least in some moments.



E.g. correspondence of peaks in two similar time series



Fixed Time Axis. Sequences are aligned “one to one”. Greatly suffers from the misalignment in data.

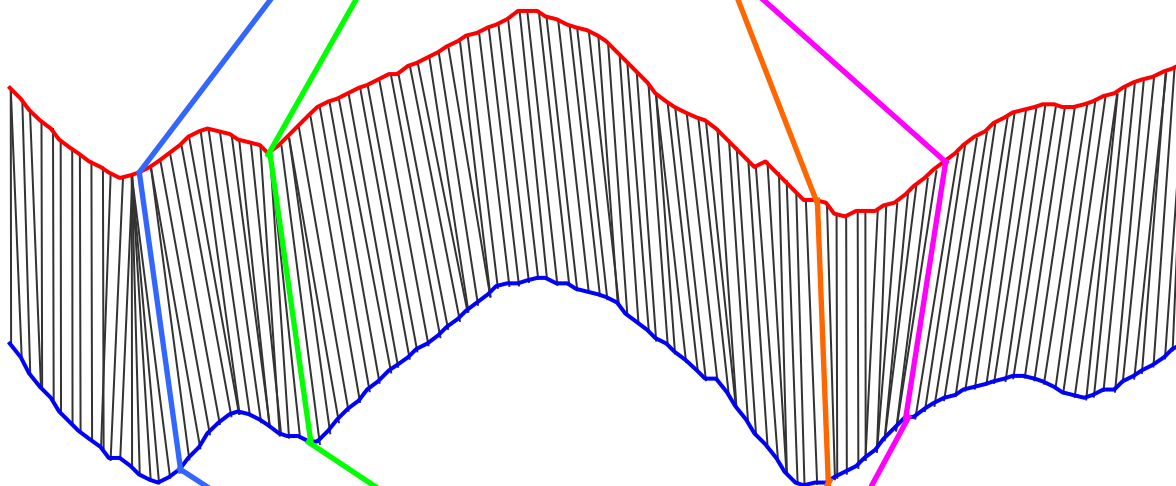
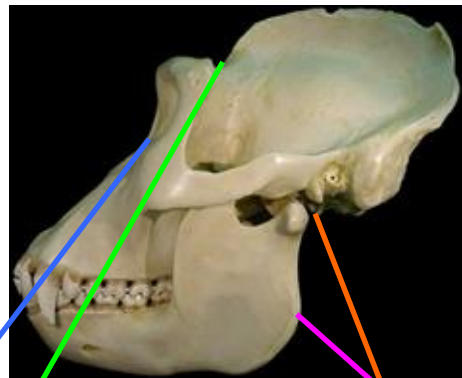


Warped Time Axis. Nonlinear alignments are possible. Can correct misalignments in data.

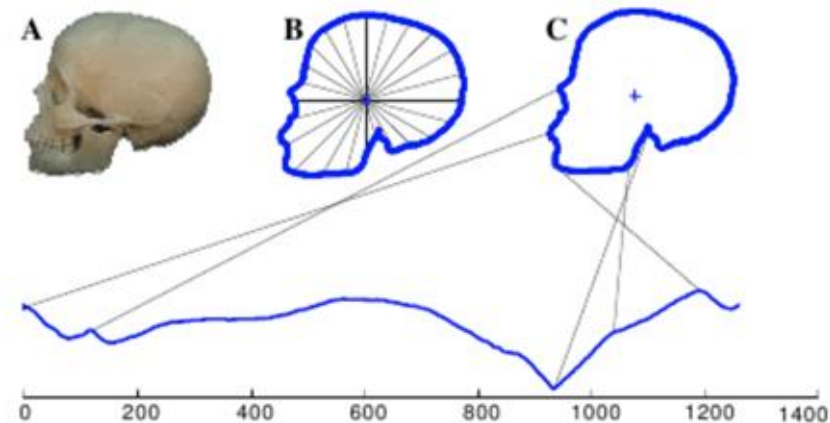
Dynamic Time Warping.

IMAGE AS A TIME SERIES

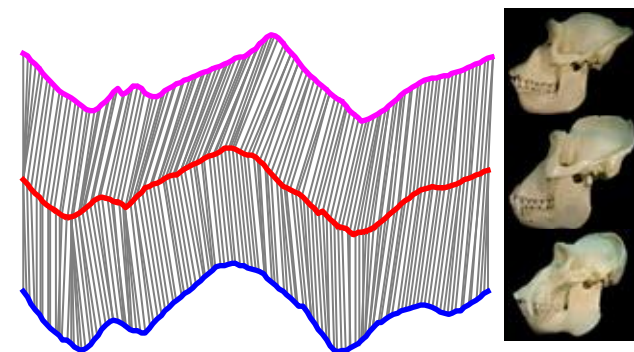
Lowland Gorilla



Mountain Gorilla

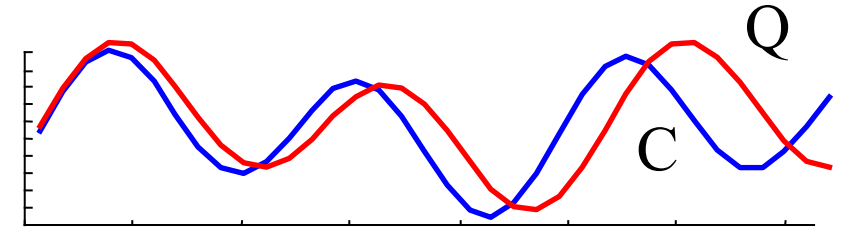
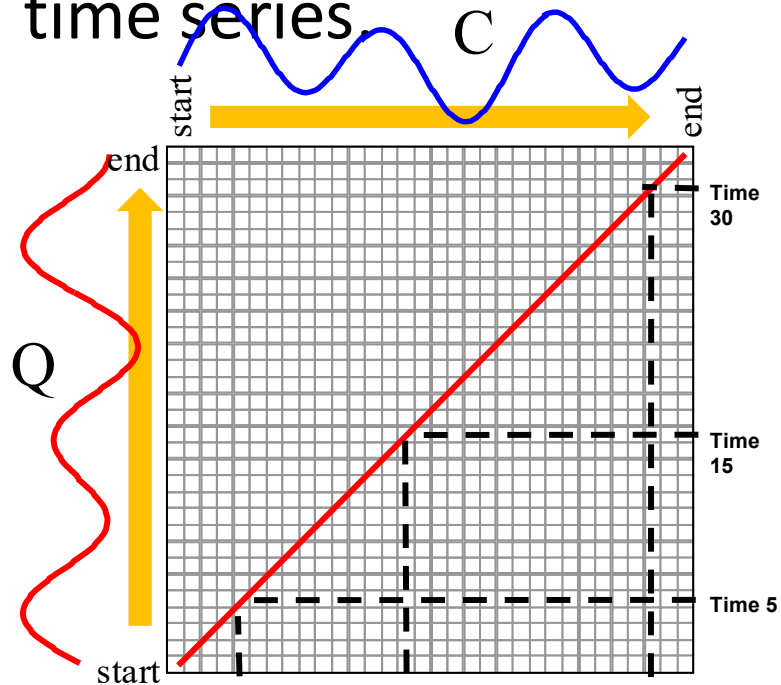


Radial Scanning



How is DTW Calculated?

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

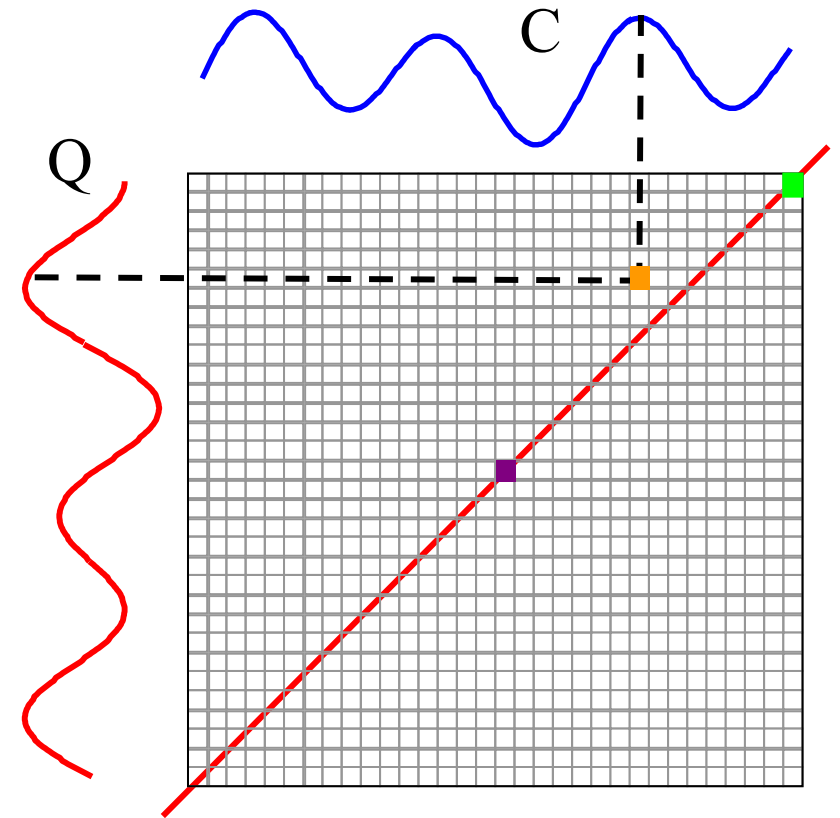
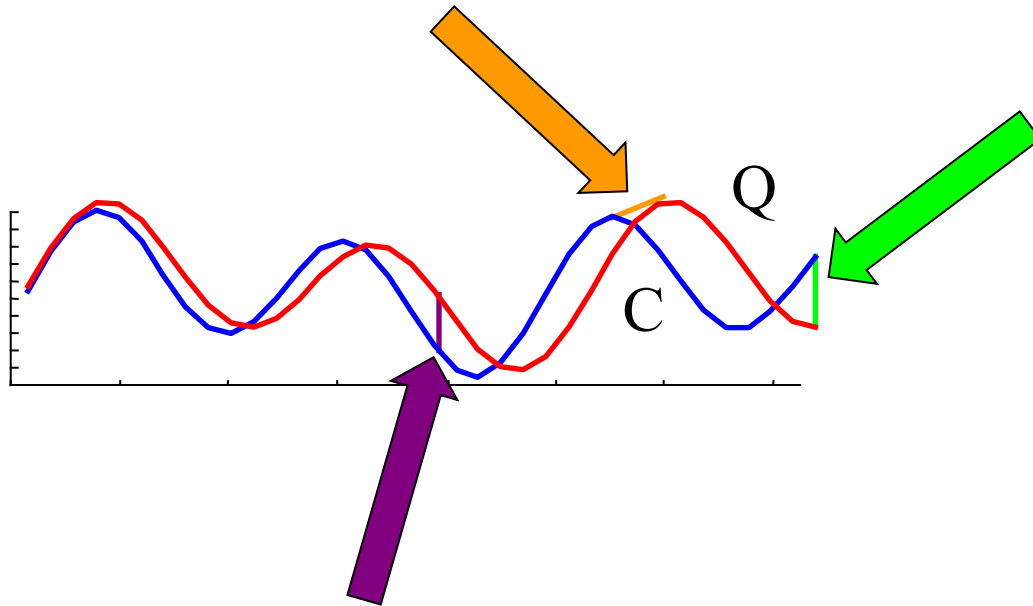


The Euclidean distance works only on the diagonal of the matrix. The sequence of comparisons performed:

- Start from pair of points $(0,0)$
- After point (i,i) move to $(i+1,i+1)$
- End the process on (n,n)

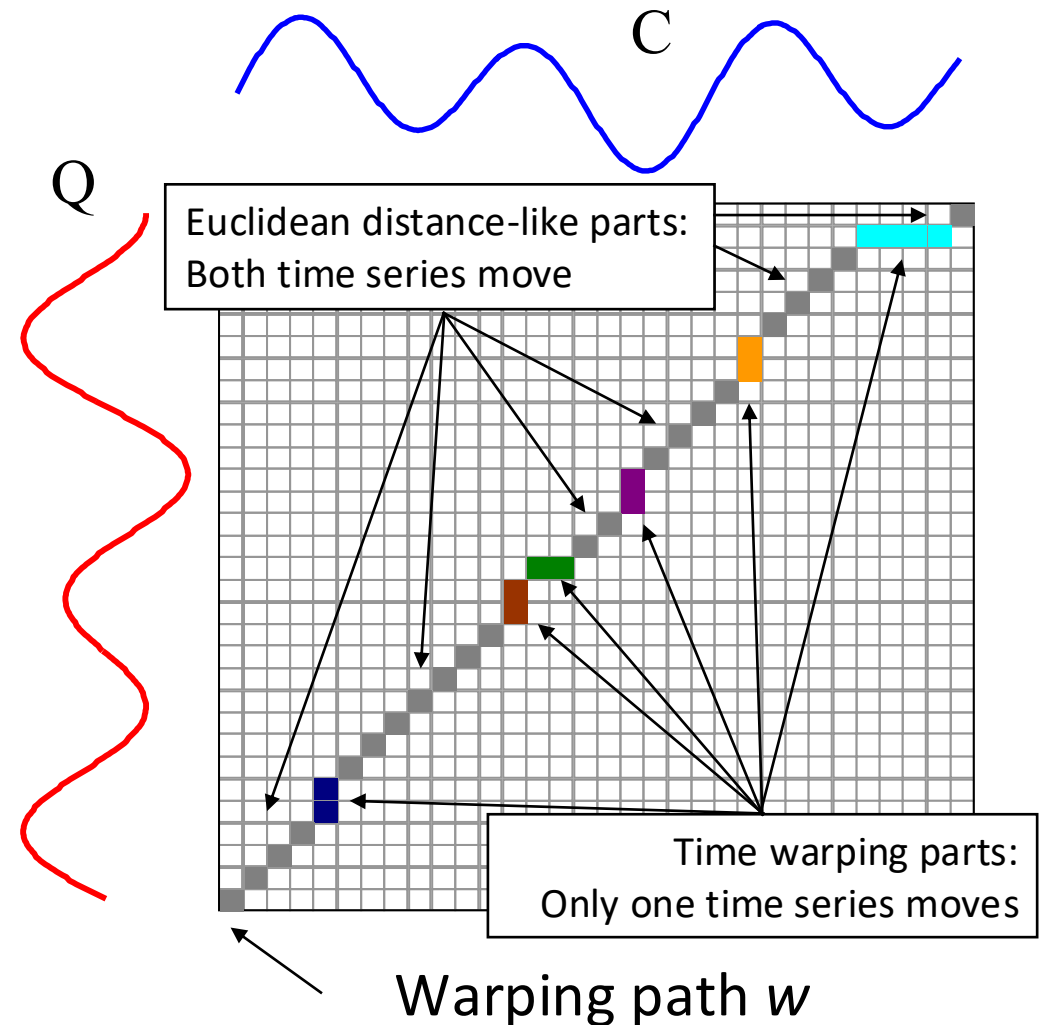
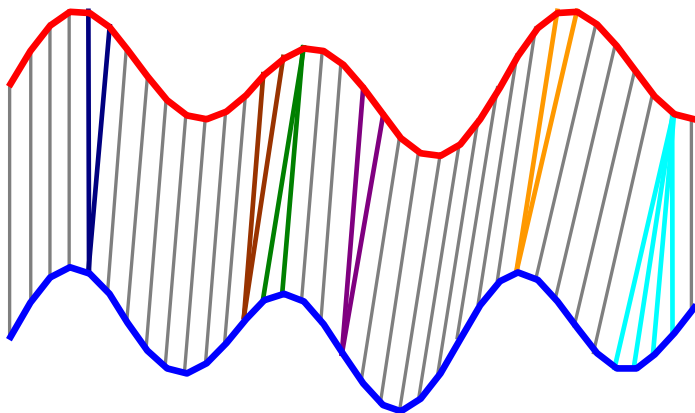
How is DTW Calculated?

- The DTW distance can “freely” move outside the diagonal of the matrix
- Such cells correspond to temporally shifted points in the two time series



How is DTW Calculated?

- Every possible warping between two time series, is a path through the matrix.
- The constrained sequence of comparisons performed:
 - Start from pair of points $(0,0)$
 - After point (i,j) , either i or j increase by one, or both of them
 - End the process on (n,n)

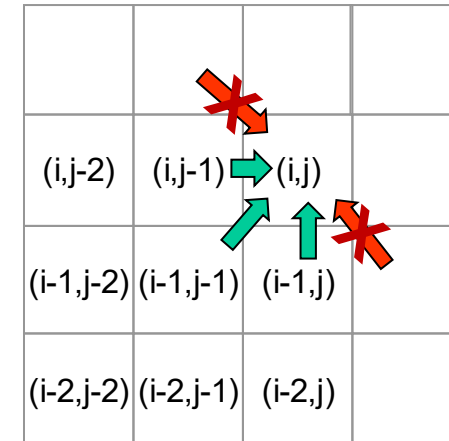


How is DTW Calculated?

- Every possible warping between two time series, is a path through the matrix.
- We find the best one using a recursive definition of the DTW:

$$\begin{aligned}\gamma(i,j) &= \text{cost of best path reaching cell } (i,j) \\ &= d(q_i, c_j) + \min \{ \gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1) \}\end{aligned}$$

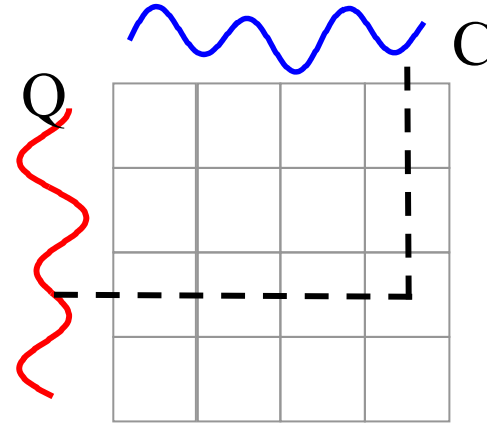
- Idea: best path must pass through $(i-1, j)$, $(i-1, j-1)$ or $(i, j-1)$



Dynamic Programming Approach

Step 1: compute the matrix of all $d(q_i, c_j)$

- Point-to-point distances $D(i,j) = |Q_i - C_j|$



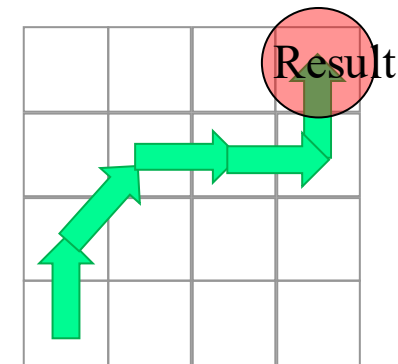
$$\gamma(i,j) = d(q_i, c_j) + \min \{ \gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1) \}$$

Step 2: compute the matrix of all path costs $\gamma(i,j)$

- Start from cell $(1,1)$
- Compute $(2,1), (3,1), \dots, (n,1)$
- Repeat for columns $2, 3, \dots, n$
- Final result in last cell computed



Step 3: find the path with the lowest value (best alignment)



Dynamic Programming Approach

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

Step 2: compute the matrix of all path costs $\gamma(i,j)$

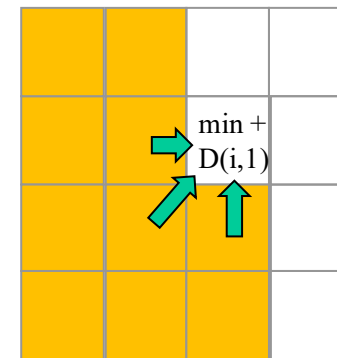
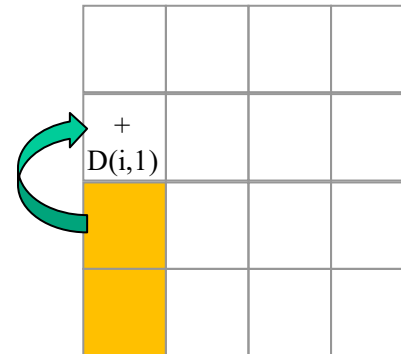
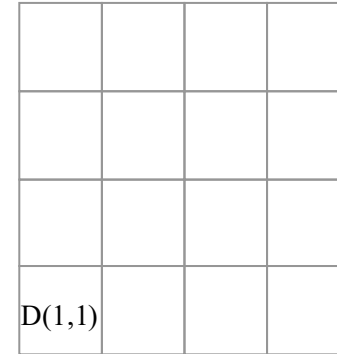
- Start from cell $(1,1)$

$$\begin{aligned} \gamma(1,1) &= d(q_1, c_1) + \min \{ \gamma(\cancel{0}, \cancel{0}), \gamma(\cancel{0}, \cancel{1}), \gamma(\cancel{1}, \cancel{0}) \} \\ &= d(q_1, c_1) \\ &= D(1,1) \end{aligned}$$

- Compute $(2,1), (3,1), \dots, (n,1)$

$$\begin{aligned} \gamma(i,1) &= d(q_i, c_1) + \min \{ \gamma(\cancel{i-1}, \cancel{0}), \gamma(i-1, 1), \gamma(\cancel{i}, \cancel{0}) \} \\ &= d(q_i, c_1) + \gamma(i-1, 1) \\ &= D(i,1) + \gamma(i-1, 1) \end{aligned}$$

- Repeat for columns $2, 3, \dots, n$
 - The general formula applies

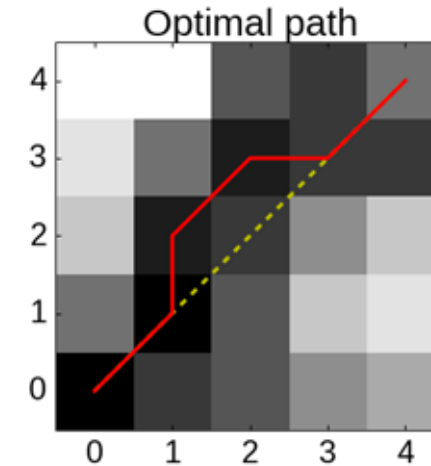
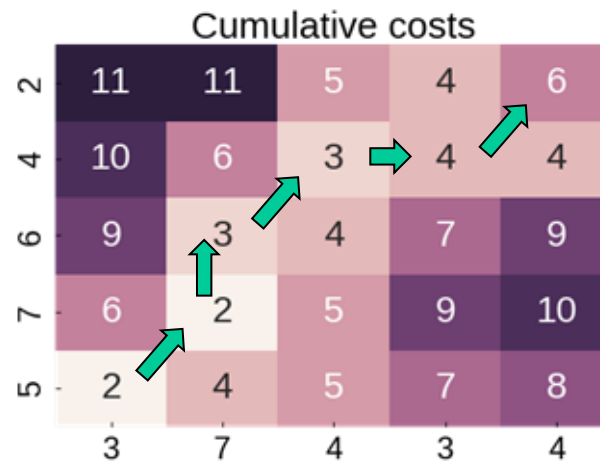
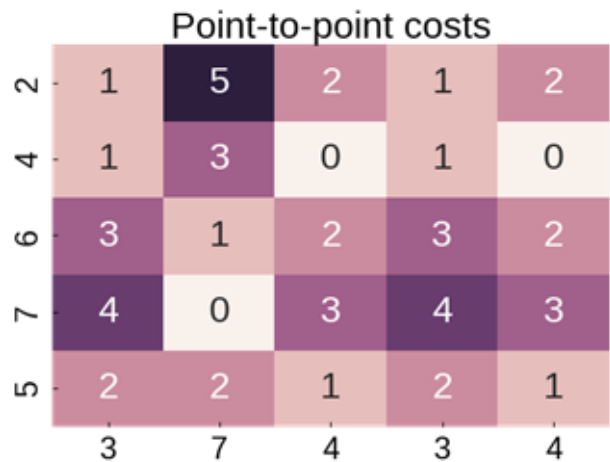


Dynamic Programming Approach

Example

- $c = \langle 3, 7, 4, 3, 4 \rangle$
- $q = \langle 5, 7, 6, 4, 2 \rangle$

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



DTW – Exercise 1

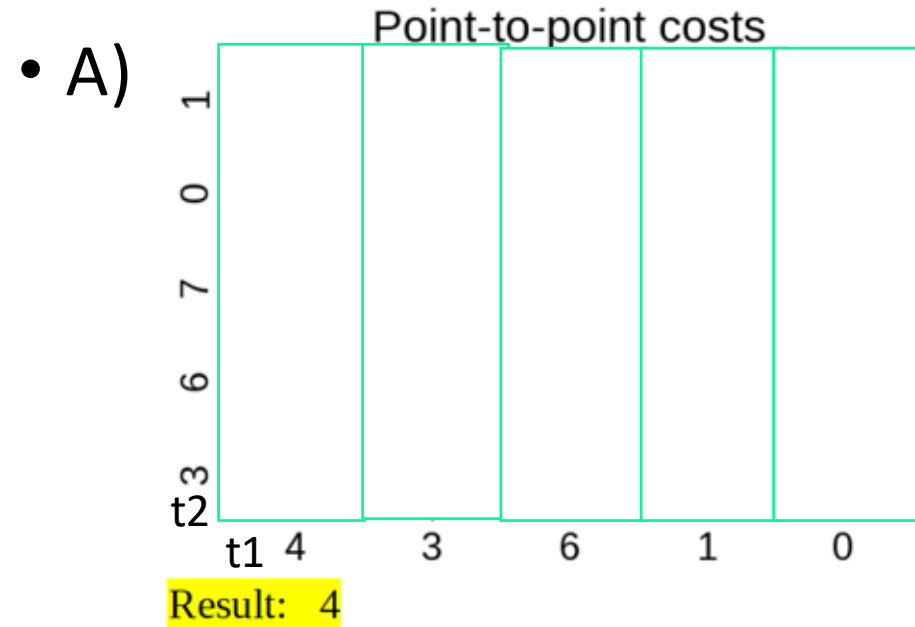
- Given the following input time series:

t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >

- A) Compute the distance between “t1” and “t2”, using the DTW with distance between points computed as $d(x,y) = |x - y|$.
- B) If we repeat the computation of point (A) above, this time with a Sakoe-Chiba band of size $r=1$, does the result change? Why?
- C) If we compute $DTW(T1, T2)$, where $T1$ is equal to $t1$ in reverse order (namely $T1 = \langle 0, 1, 6, 3, 4 \rangle$) and similarly for $T2$ (namely $T2 = \langle 1, 0, 7, 6, 3 \rangle$), is it true that $DTW(T1, T2) = DTW(t1, t2)$? Discuss the problem without providing any computation.

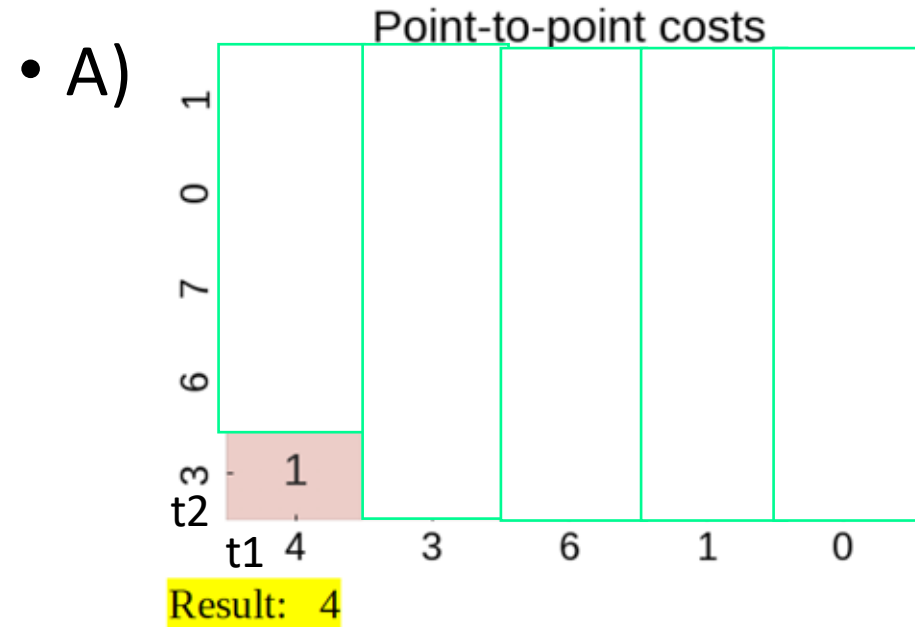
DTW – Exercise 1 - Solution

t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >



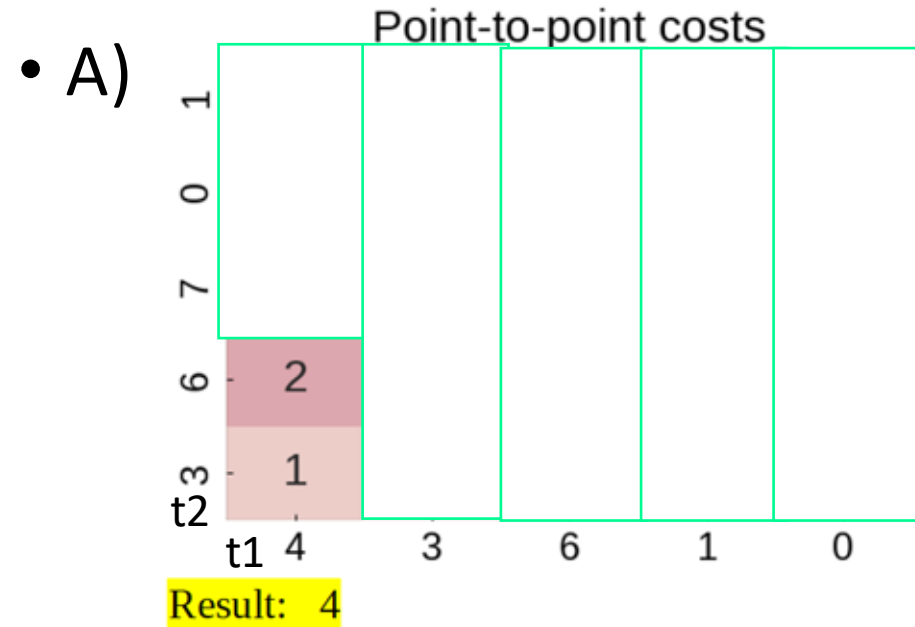
DTW – Exercise 1 - Solution

t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >



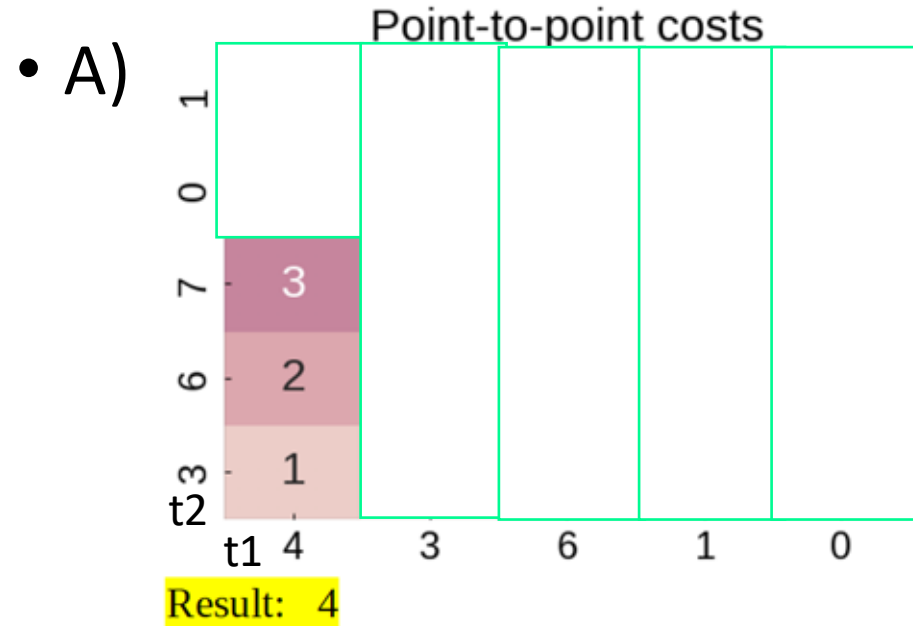
DTW – Exercise 1 - Solution

t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >



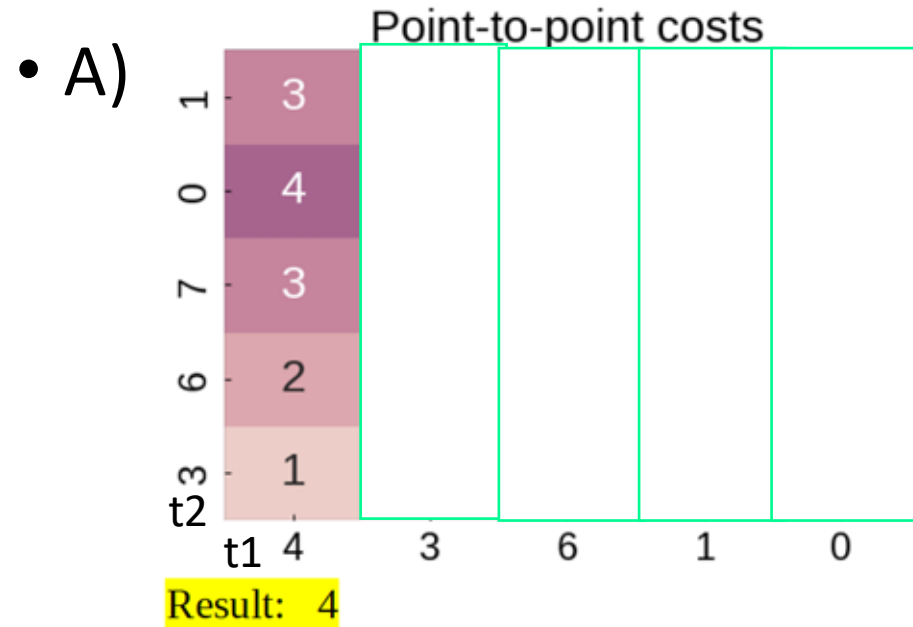
DTW – Exercise 1 - Solution

t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >



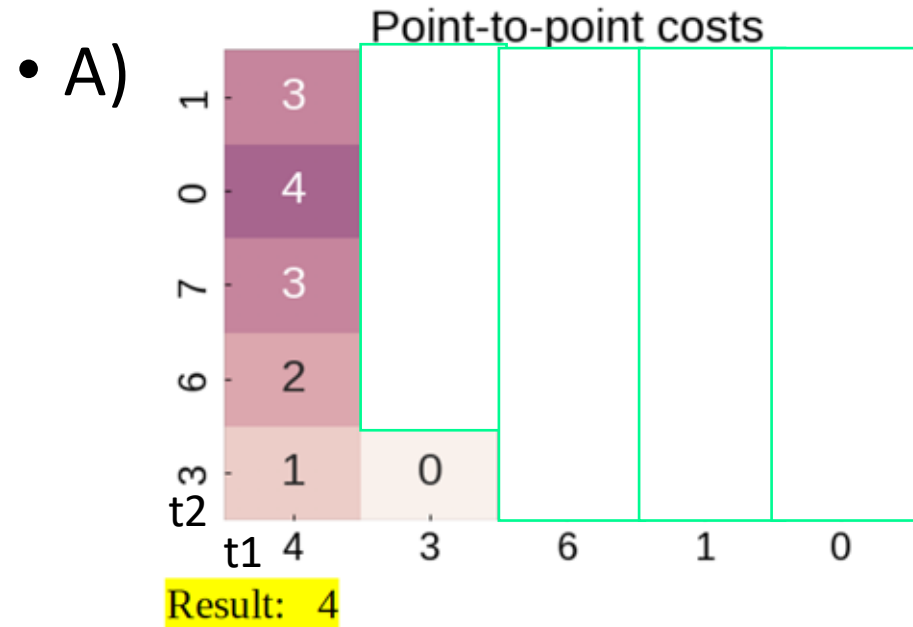
DTW – Exercise 1 - Solution

t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >



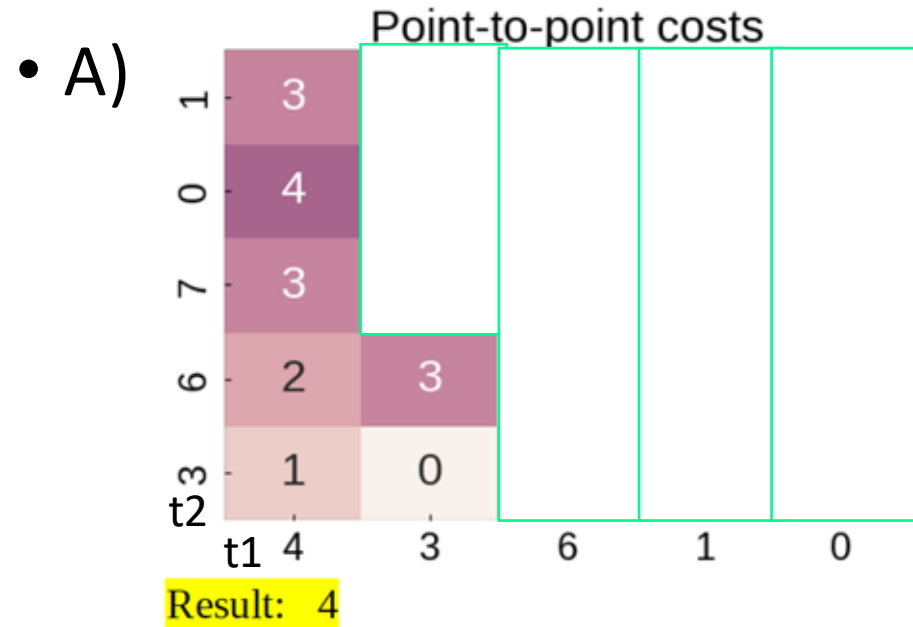
DTW – Exercise 1 - Solution

t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >



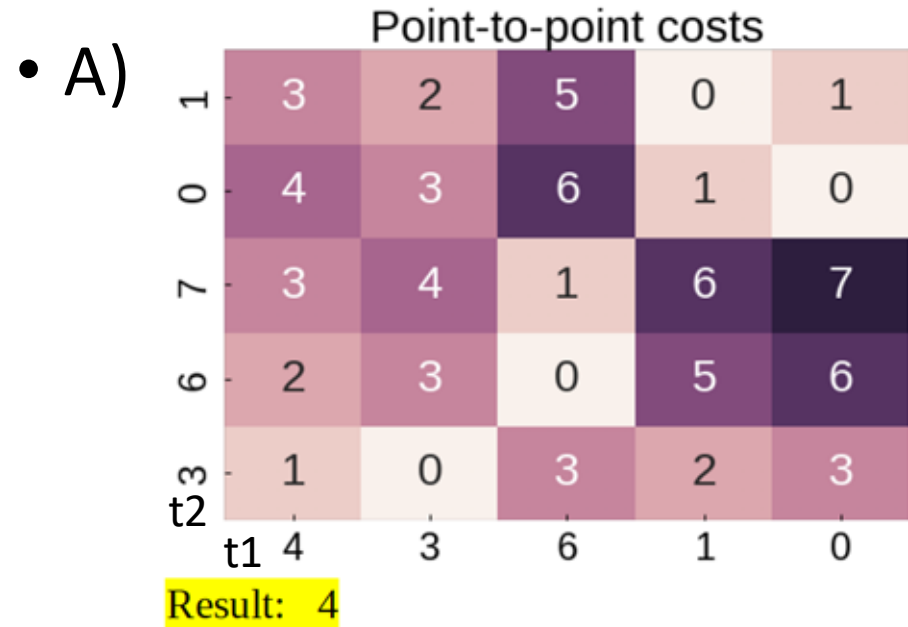
DTW – Exercise 1 - Solution

t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >



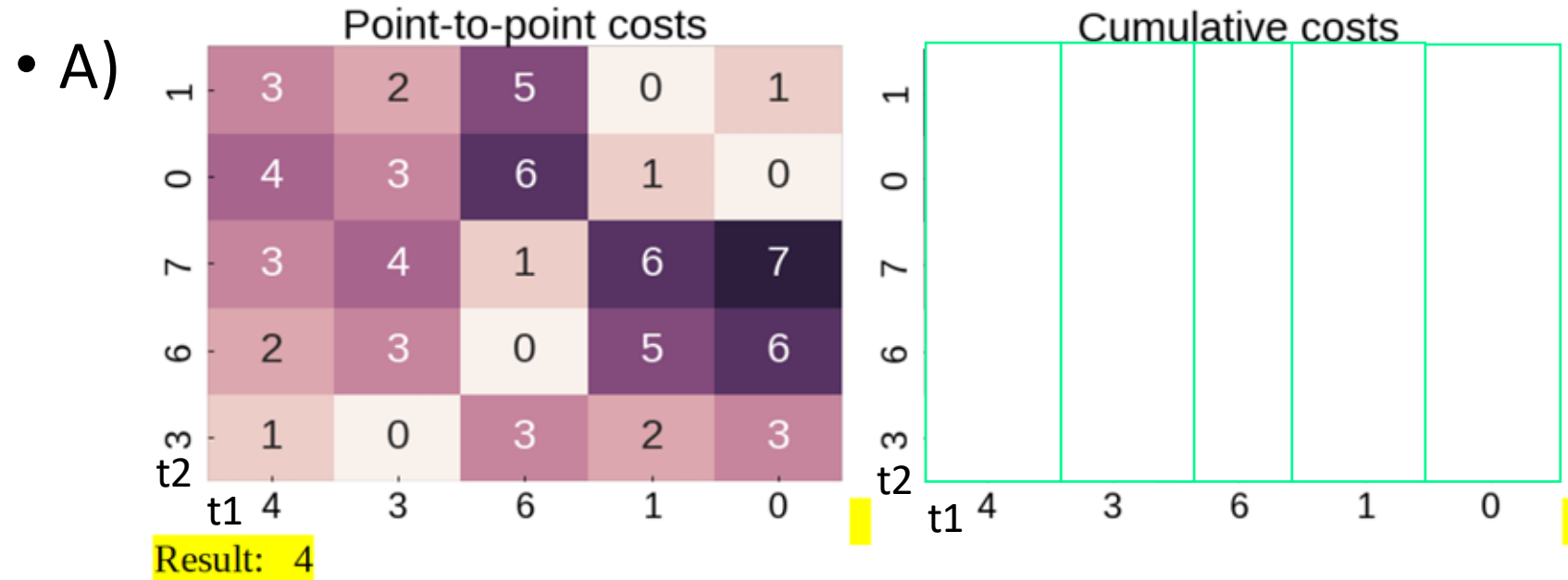
DTW – Exercise 1 - Solution

t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >



DTW – Exercise 1 - Solution

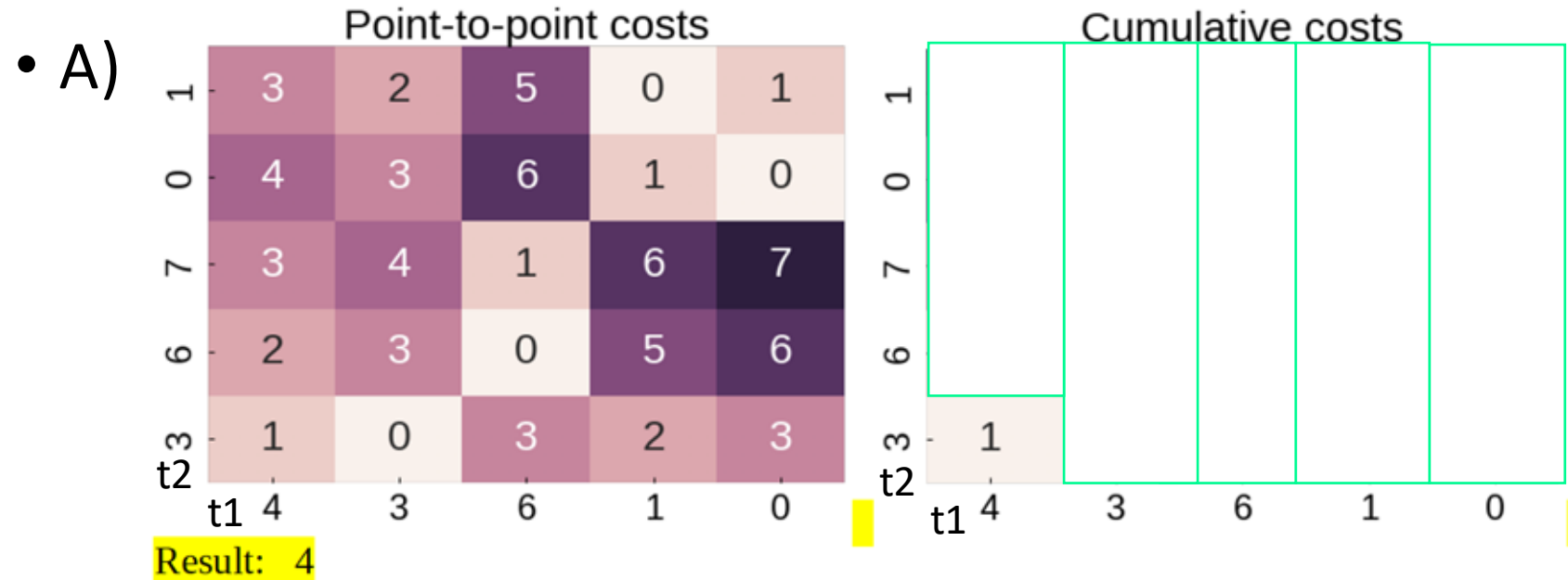
t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >



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

DTW – Exercise 1 - Solution

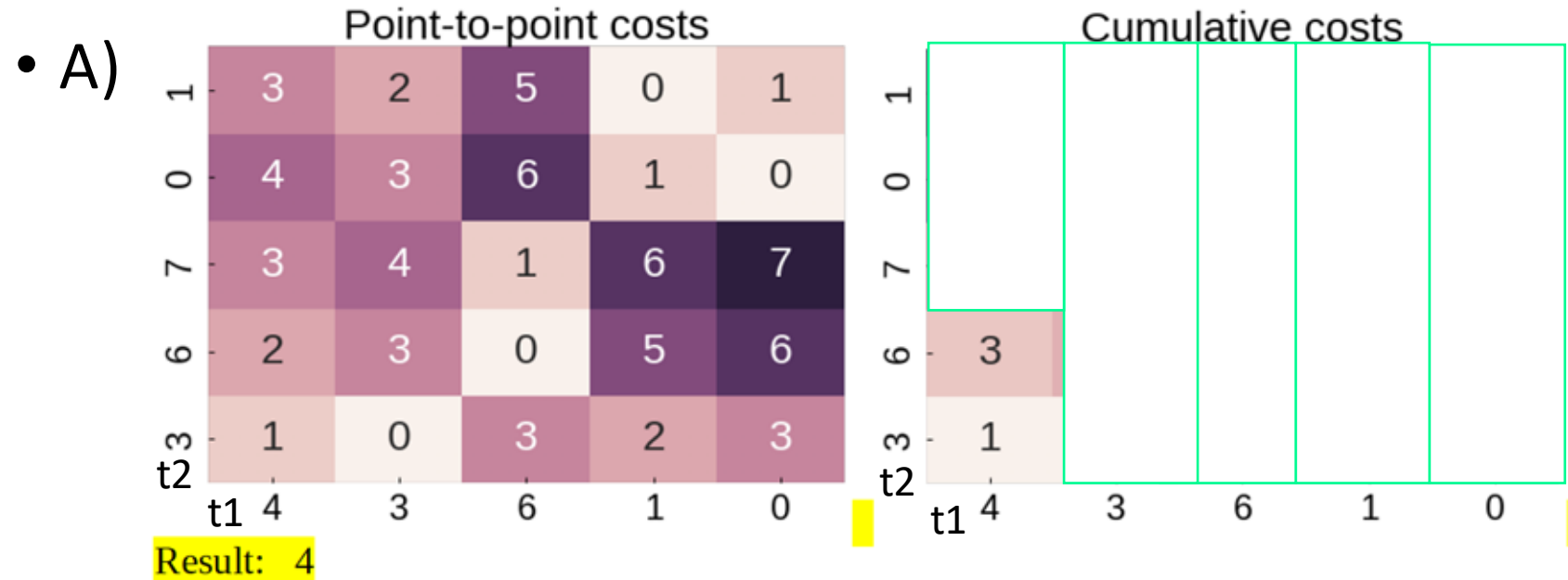
t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >



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

DTW – Exercise 1 - Solution

t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >



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

DTW – Exercise 1 - Solution

t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >

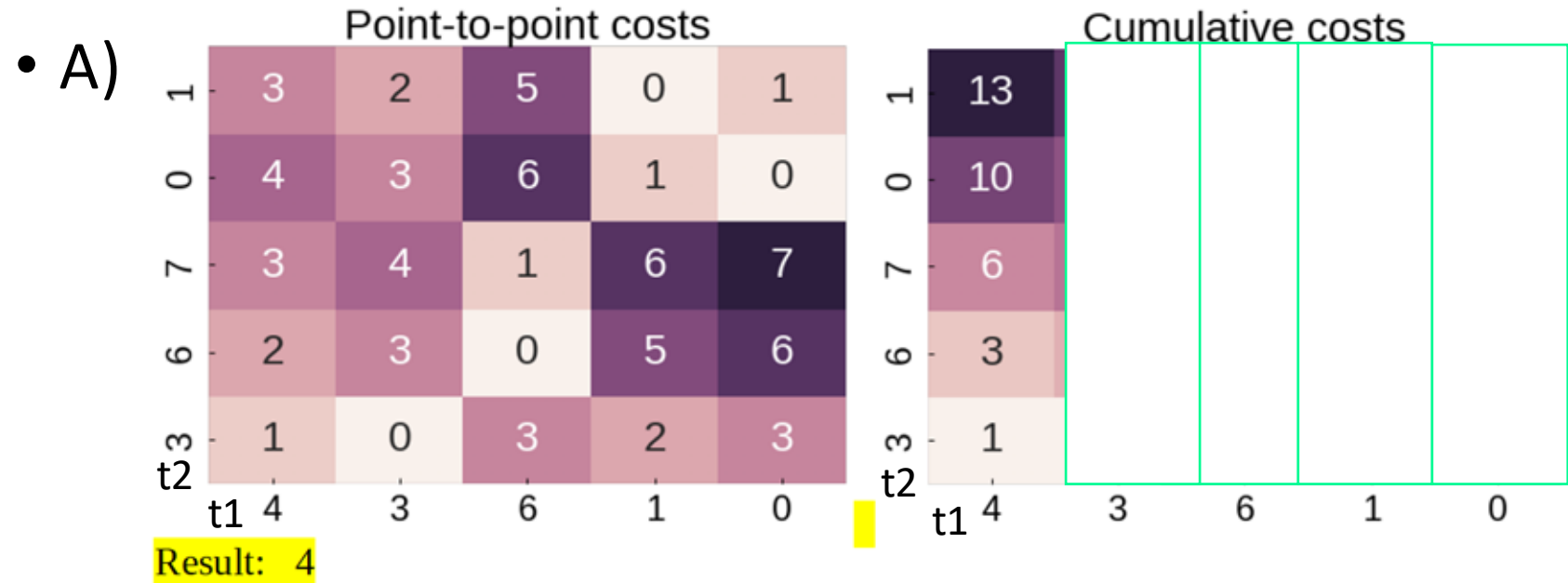
• A)



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

DTW – Exercise 1 - Solution

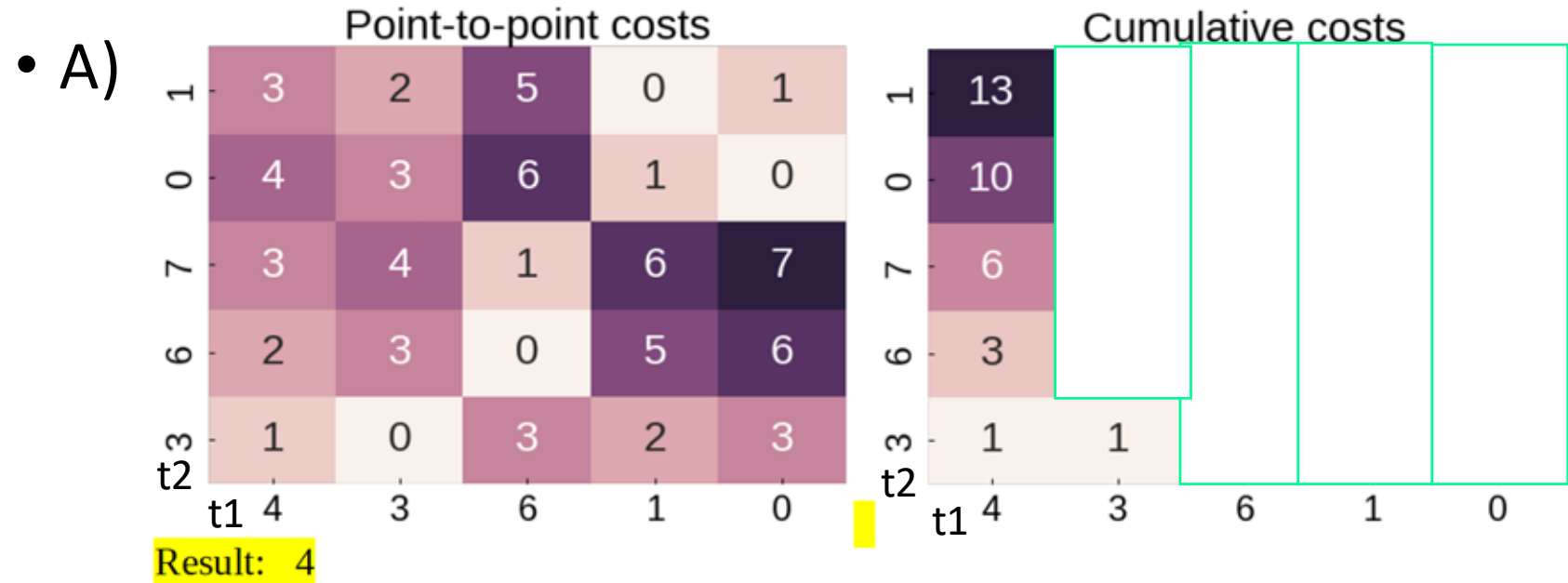
t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >



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

DTW – Exercise 1 - Solution

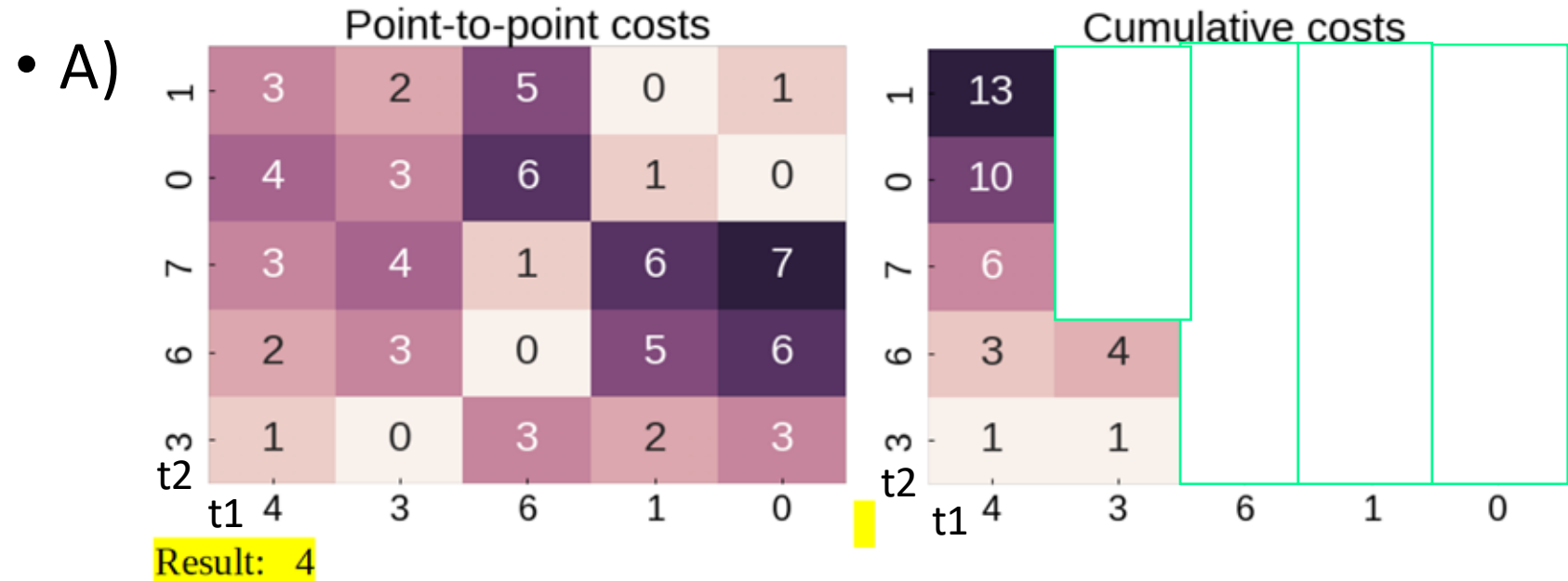
t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >



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

DTW – Exercise 1 - Solution

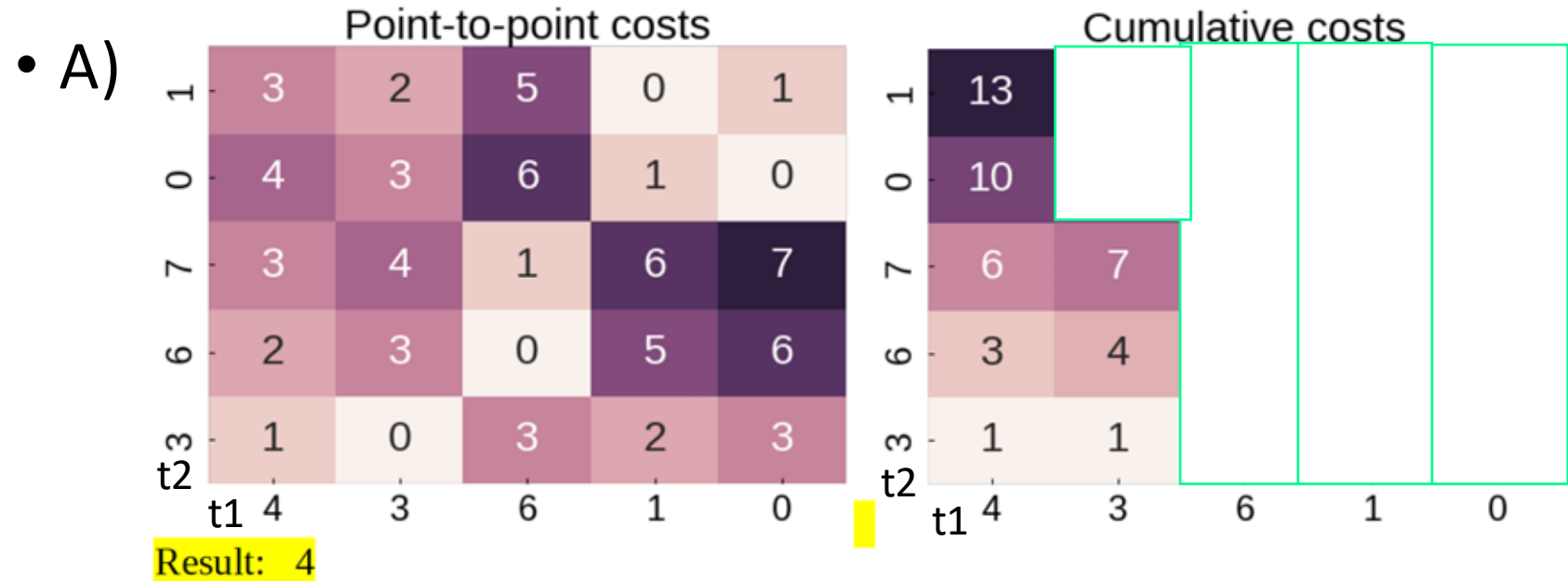
t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >



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

DTW – Exercise 1 - Solution

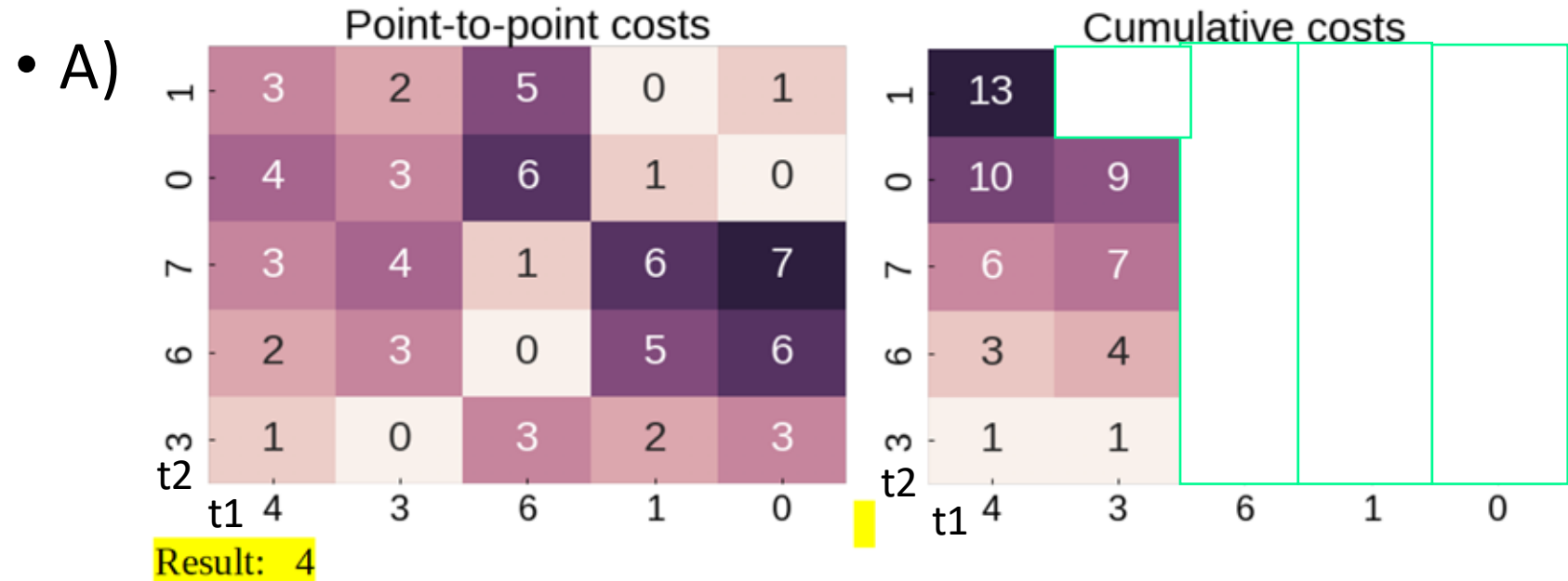
t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >



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

DTW – Exercise 1 - Solution

t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >

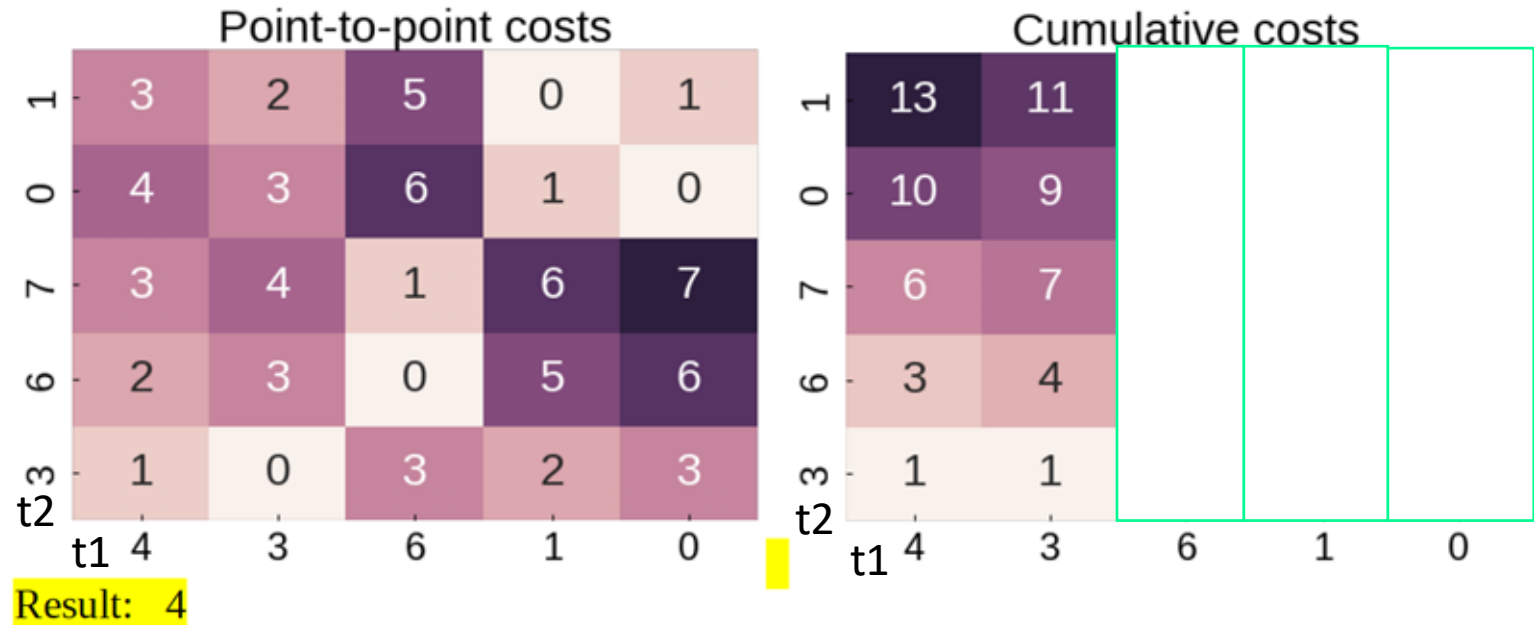


$$\gamma(i,j) = d(q_i, c_j) + \min \{ \gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1) \}$$

DTW – Exercise 1 - Solution

t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >

• A)

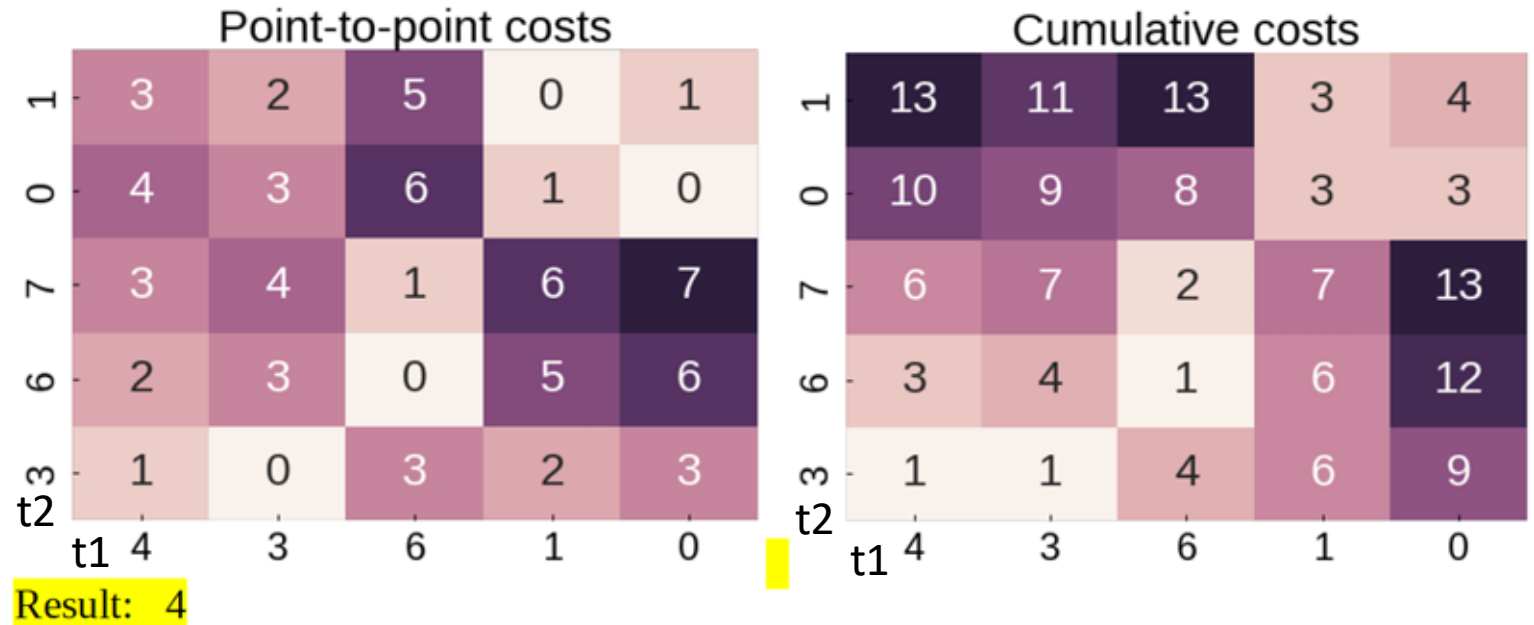


$$\gamma(i,j) = d(q_i, c_j) + \min \{ \gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1) \}$$

DTW – Exercise 1 - Solution

t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >

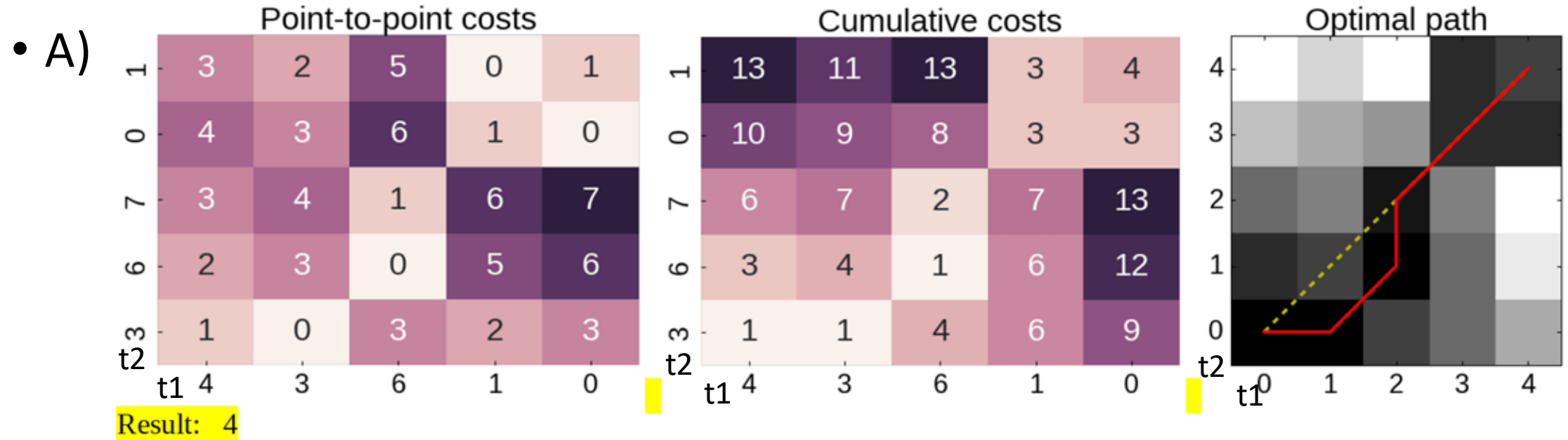
• A)



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

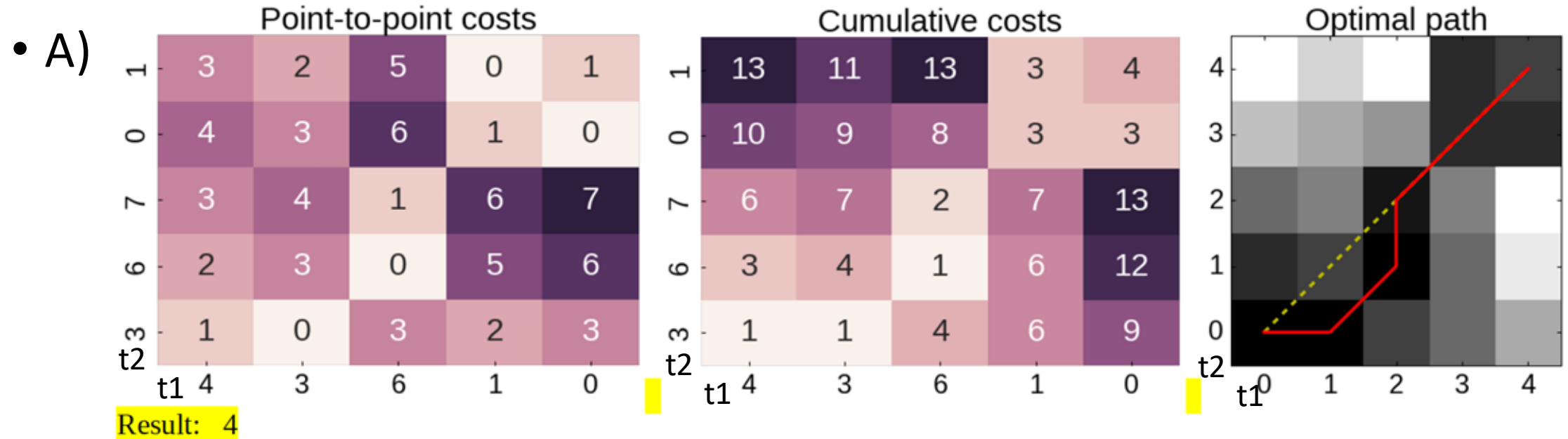
DTW – Exercise 1 - Solution

t1	$\langle 4, 3, 6, 1, 0 \rangle$
t2	$\langle 3, 6, 7, 0, 1 \rangle$



DTW – Exercise 1 - Solution

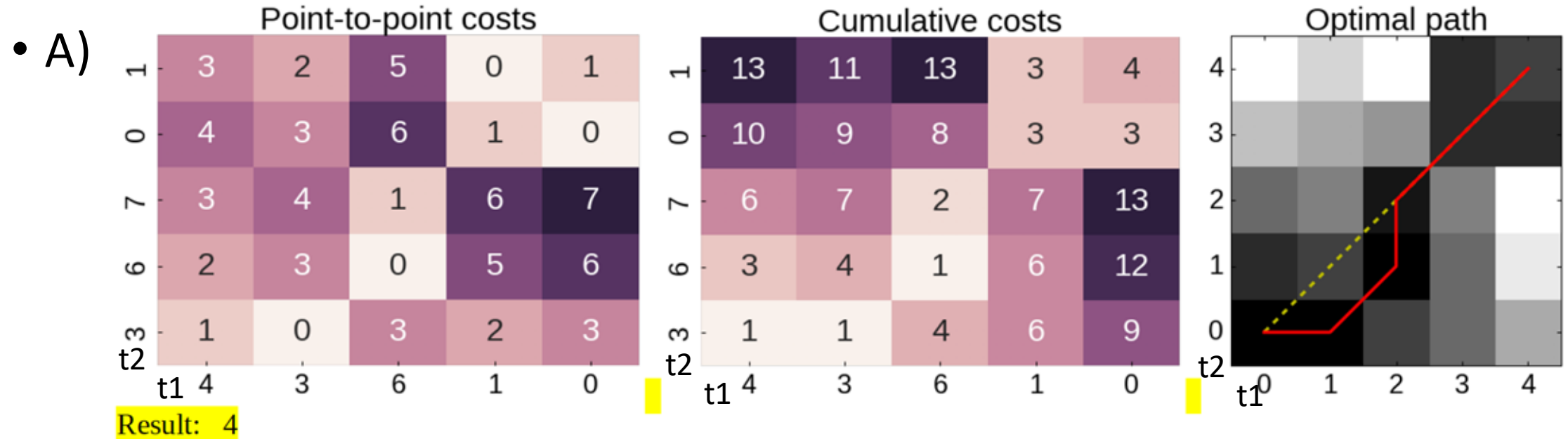
t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >



- B) No. Because the DTW optimal path remains inside the band of size $r=1$

DTW – Exercise 1 - Solution

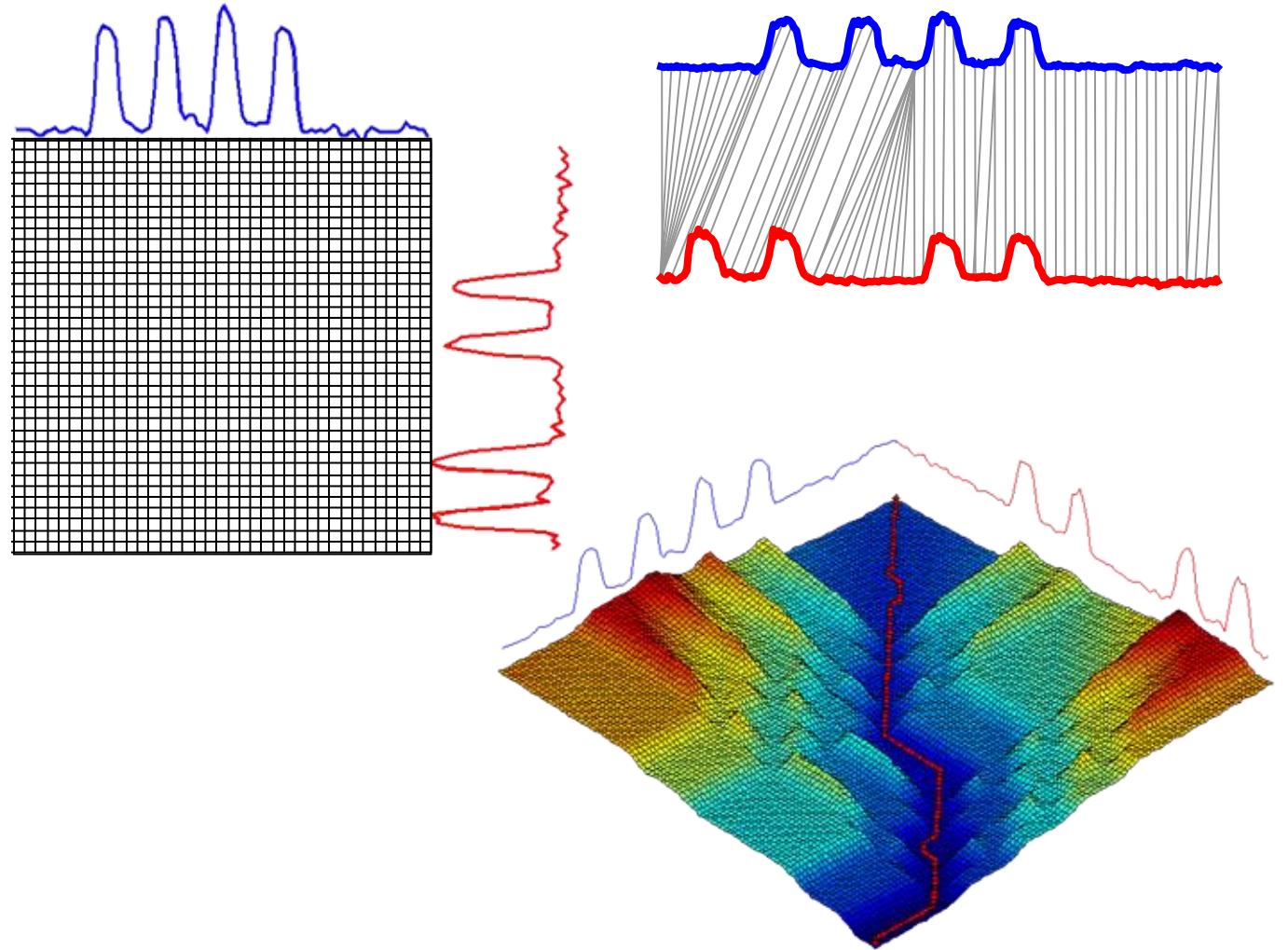
t1	< 4, 3, 6, 1, 0 >
t2	< 3, 6, 7, 0, 1 >



- B) No. Because the DTW optimal path remains inside the band of size $r=1$
- C) Yes. The optimal path in one direction is the same in the opposite direction. Though, the cumulative costs matrix might look different.

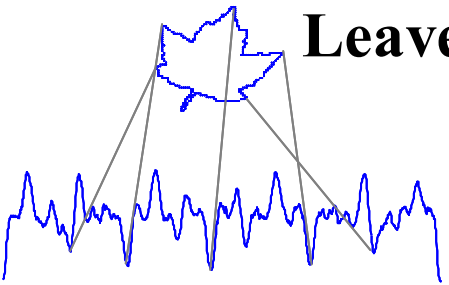
Dynamic Time Warping – A Real Example

- A Real Example
- 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.



Comparison of Euclidean Distance and DTW

Leaves



Faces

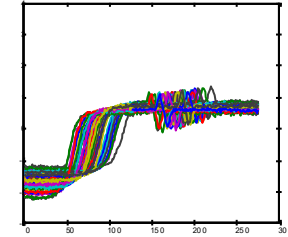


I SHOW YOU.

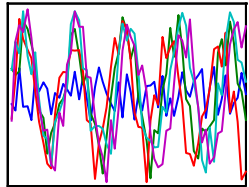
YOU SHOW ME.



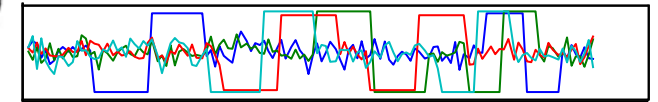
Trace



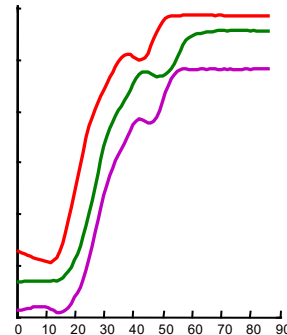
Control



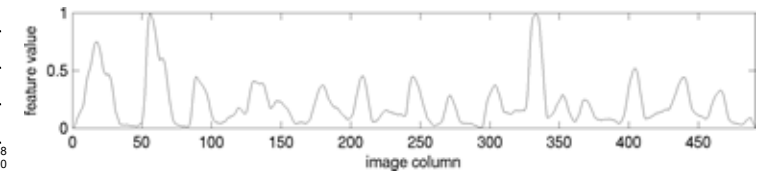
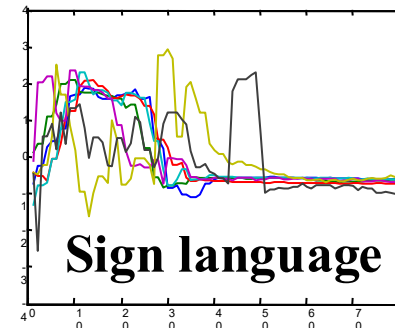
2-Patterns



Gun



Sign language



Word Spotting

Comparison of Euclidean Distance and DTW

- Classification using 1-NN
- $\text{Class}(x)$ = class of most similar training object
- Leaving-one-out evaluation
- For each object: use it as test set, return overall average

Error Rate

Dataset	Euclidean	DTW
Word Spotting	4.78	1.10
Sign language	28.70	25.93
GUN	5.50	1.00
Nuclear Trace	11.00	0.00
Leaves [#]	33.26	4.07
(4) Faces	6.25	2.68
Control Chart*	7.5	0.33
2-Patterns	1.04	0.00

Comparison of Euclidean Distance and DTW

- Classification using 1-NN
- $\text{Class}(x)$ = class of most similar training object
- Leaving-one-out evaluation
- For each object: use it as test set, return overall average
- DTW is two to three orders of magnitude slower than Euclidean distance.

Milliseconds

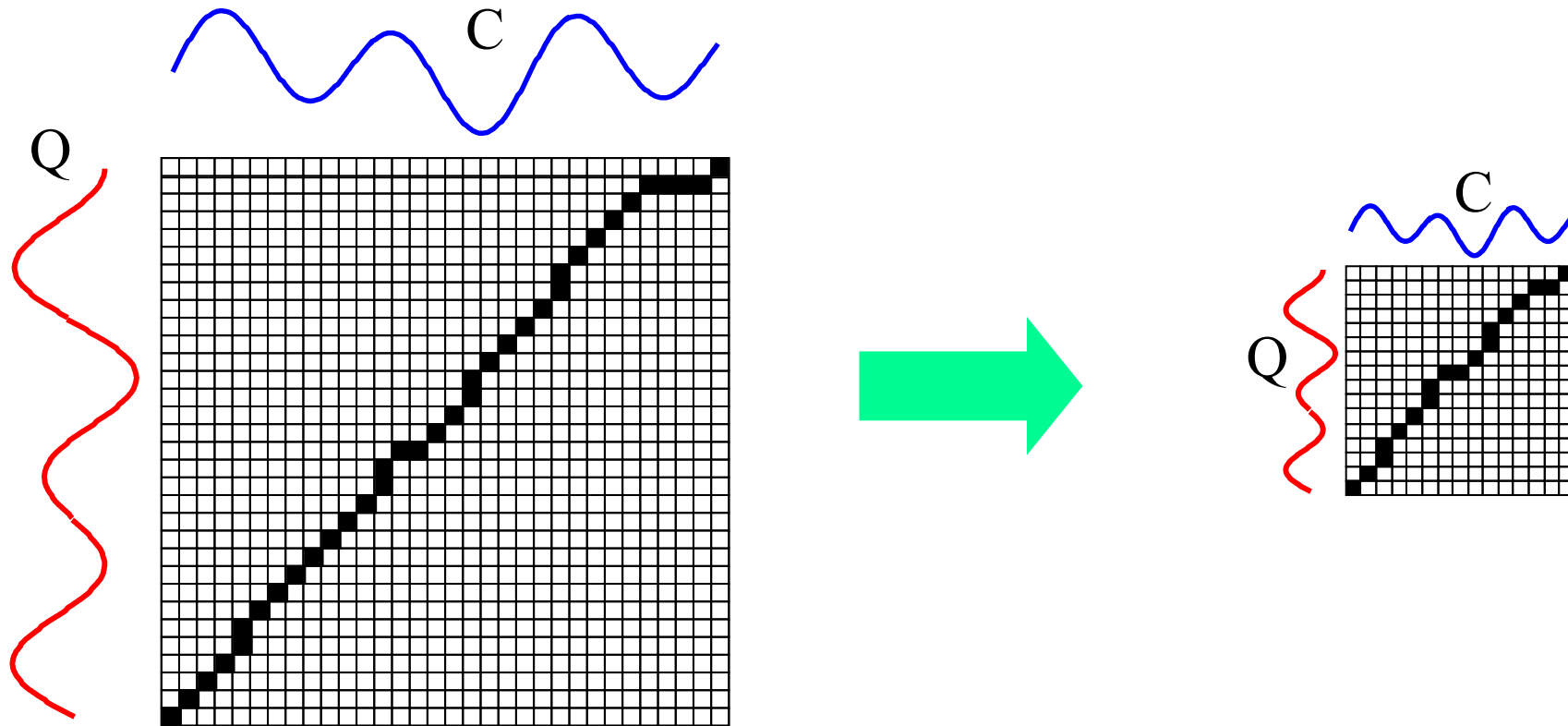
Dataset	Euclidean	DTW
Word Spotting	40	8,600
Sign language	10	1,110
GUN	60	11,820
Nuclear Trace	210	144,470
Leaves	150	51,830
(4) Faces	50	45,080
Control Chart	110	21,900
2-Patterns	16,890	545,123

What we have seen so far...

- Dynamic Time Warping gives much better results than Euclidean distance on many problems.
- Dynamic Time Warping is very very slow to calculate!
- Is there anything we can do to speed up similarity search under DTW?

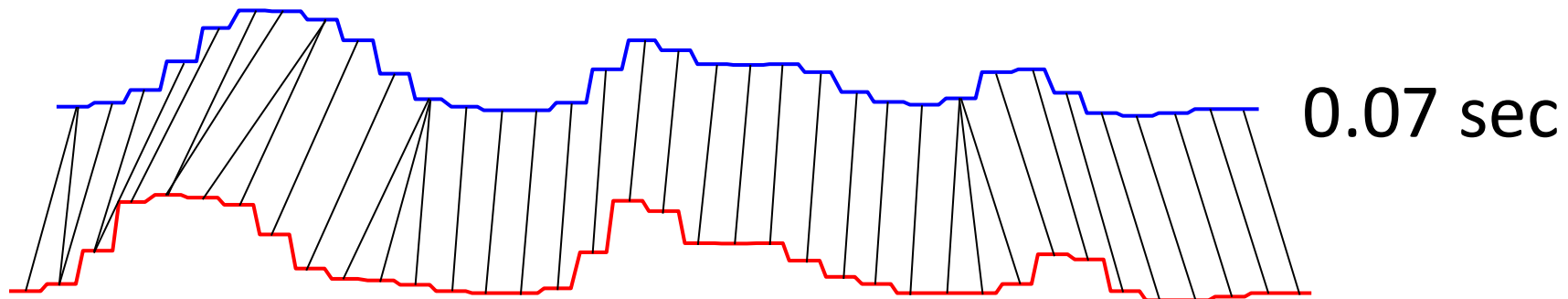
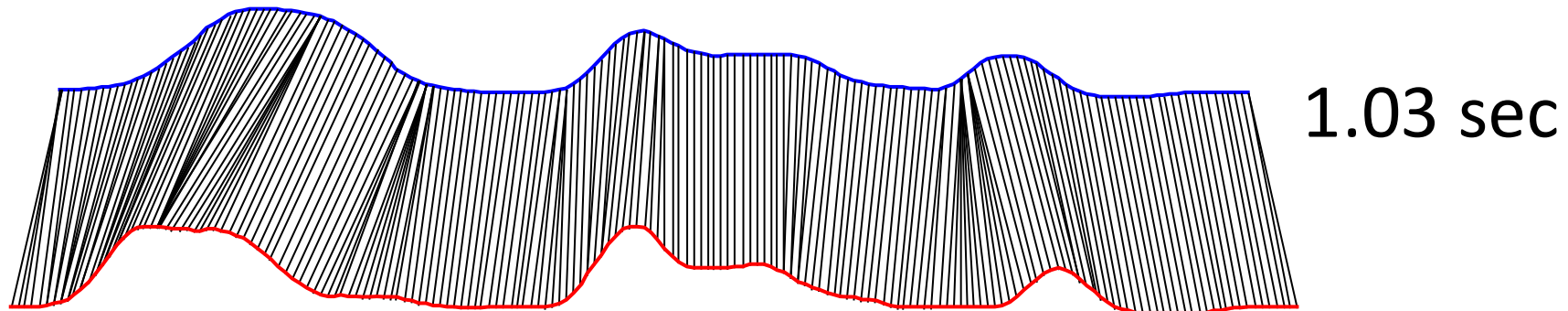
Fast Approximations to DTW

- Approximate the time series with some compressed or downsampled representation, and do DTW on the new representation.



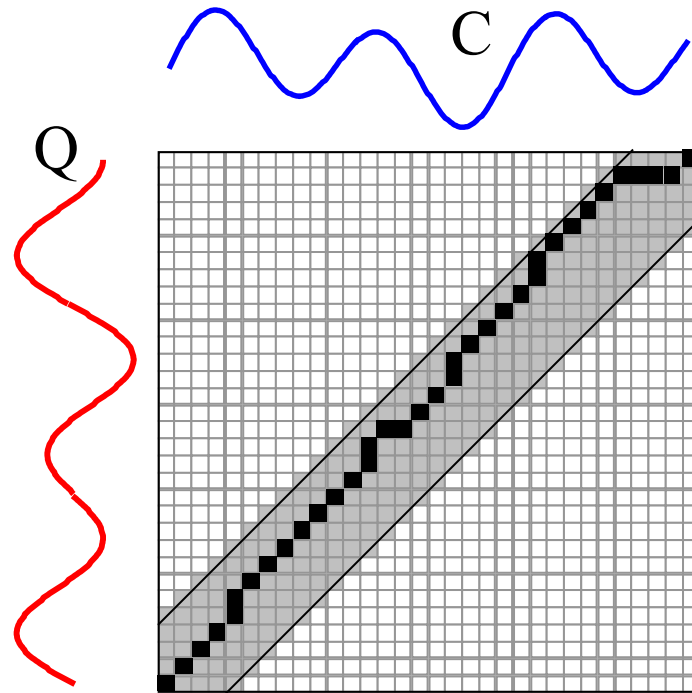
Fast Approximations to DTW

- There is strong visual evidence to suggest it works well
- In the literature there is good experimental evidence for the utility of the approach on clustering, classification, etc.

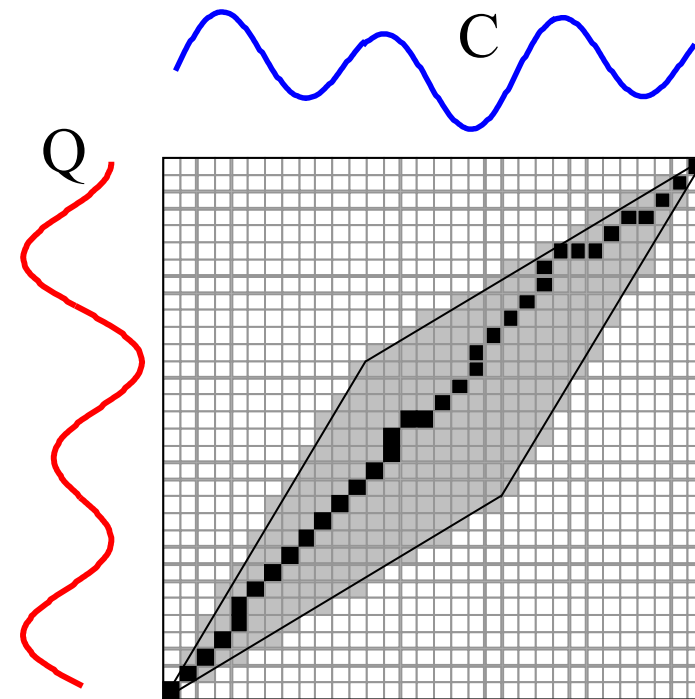


Global Constraints

- Slightly speed up the calculations
- Prevent pathological warpings



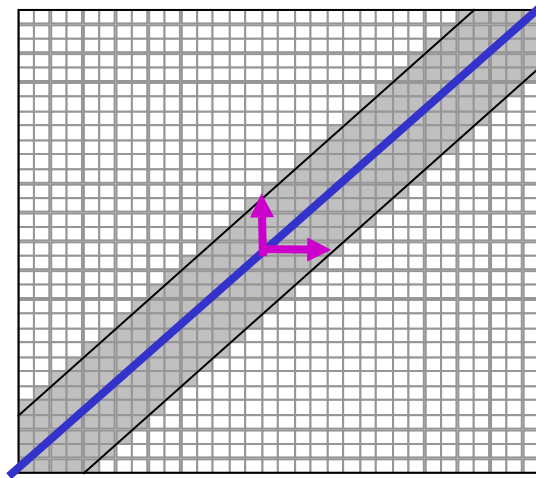
Sakoe-Chiba
Band



Itakura
Parallelogram

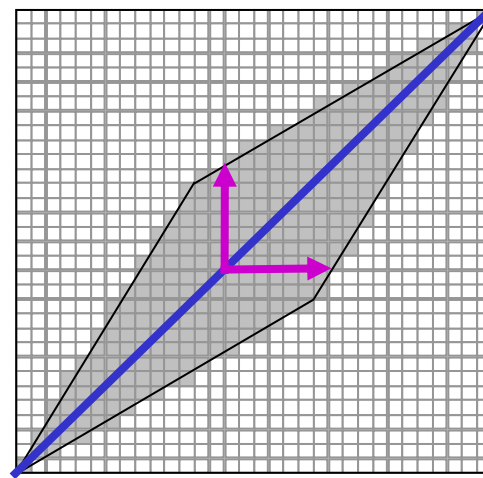
Global Constraints

- A global constraint constrains the indices of the warping path $w_k = (i, j)_k$ such that $j-r \leq i \leq j+r$, where r is a term defining allowed range of warping for a given point in a sequence.
- r can be considered as a *window* that reduces the number of calculus.



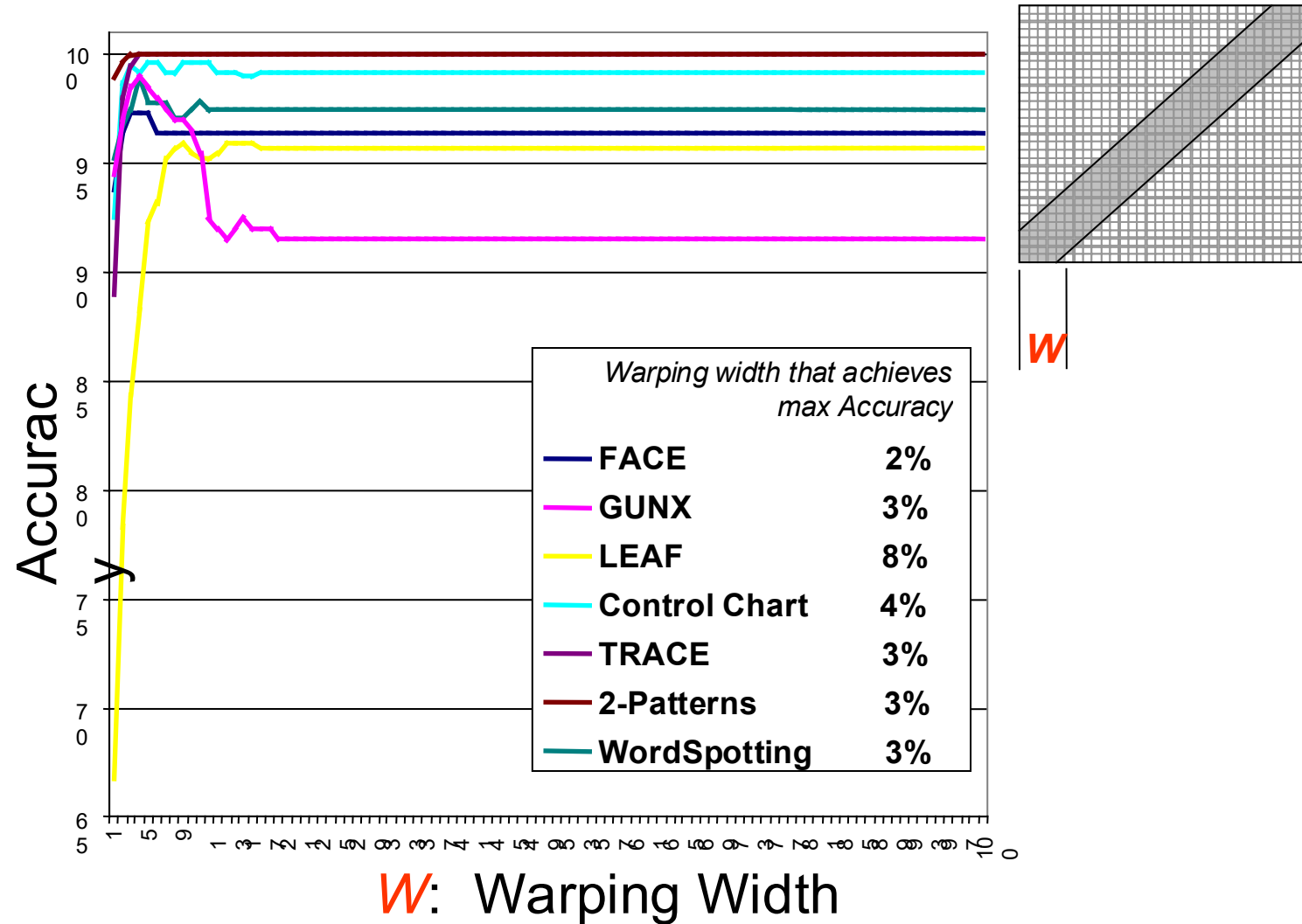
Sakoe-Chiba
Band

r_i



Itakura
Parallelogram

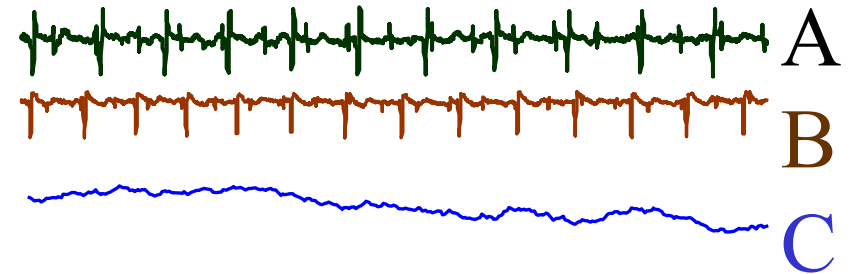
Accuracy vs. Width of Warping Window



Structural-based Similarities

Structure or Model Based Similarity

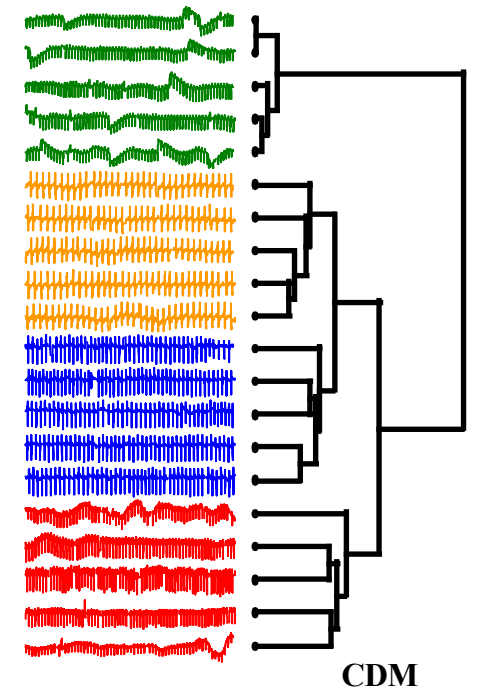
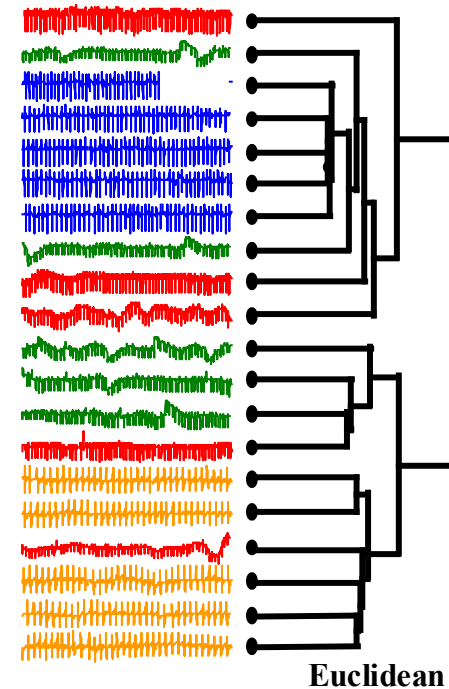
- For long time series, shape-based similarity give very poor results.
- We need to measure similarity based on high level structure.
- The basic idea is to:
 1. extract *global* features from the time series,
 2. create a feature vector, and
 3. use it to measure similarity and/or classify
- Example of features:
 - mean, variance, skewness, kurtosis,
 - 1st derivative mean, 1st derivative variance, ...
 - parameters of regression, forecasting, Markov model



Feature\Time Series	A	B	C
Max Value	11	12	19
Mean	5.3	6.4	4.8
Min Value	3	2	5
Autocorrelation	0.2	0.3	0.5
...	...	...	...

Compression Based Dissimilarity

- Use as features whatever structure a given compression algorithm finds.
- $d(x, y) = CDM(x, y) = \frac{C(x, y)}{C(x) + C(y)}$
- Time series can be compressed using various transformations:
 - Piecewise Linear Approximation
 - Adaptive Piecewise Constant Approximation
 - Symbolic Aggregate Approximation



Summary of Time Series Similarity

- If you have short time series
 - use DTW after searching over the warping window size
- If you have long time series
 - if you do know something about your data => extract features
 - and you know nothing about your data => try compression/approximation based dissimilarity

Time Series Approximation

Time Series Approximation

- **Approximation:** represent a TS into a new smaller and simpler space and use this novel representation for computing.
- **Approximation** is a special form of **Dimensionality Reduction** specifically designed for TSs.
- **Approximation vs Compression:**
 - the approximated space is always understandable
 - the compressed space is not necessarily understandable.

Time Series Approximation

Approximated representations can be

- **Time-Dependent:** the approximated values maintain a temporal ordering
- **Time-Independent:** the approximated values loose the temporal ordering
- **Instance-wise:** the approximation operation involves only the values of a single TS
- **Dataset-wise:** the approximation operation involves the values of a dataset of TS

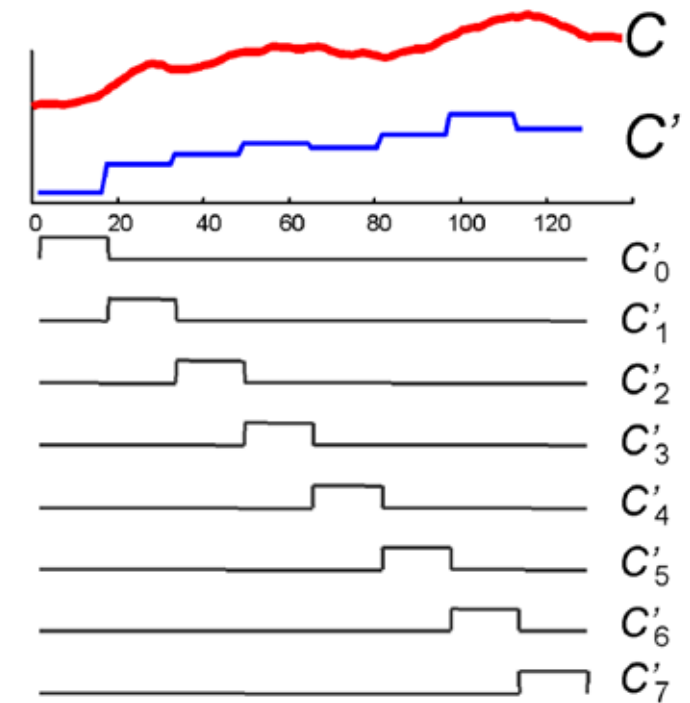
Piecewise Aggregate Approximation (PAA)

- Represent the time series as a sequence of box basis functions with each box of the same size.
- It approximates a TS by dividing it into equal-length segments and recording the **mean value of the data points** that fall within the segment.
- It reduces the data from n dimensions to M dimensions by dividing the time series into M equi-sized “frames”.
- The **mean value of the data falling within a frame** is calculated, and a vector of these values becomes the data reduced representation.

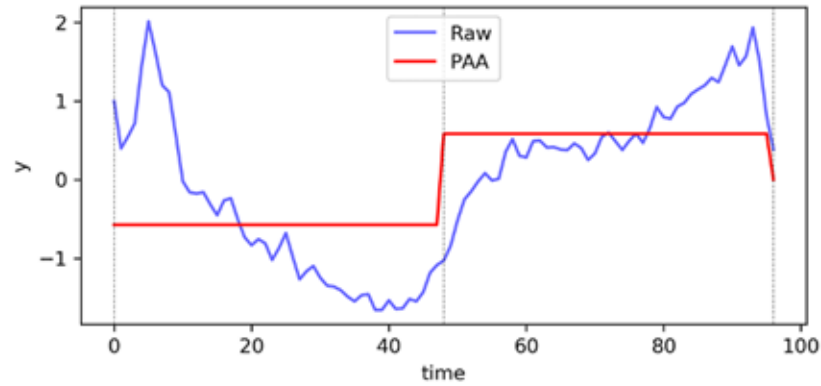
- Pros

- Extremely fast to calculate
- Supports non Euclidean measures
- Supports weighted Euclidean distance

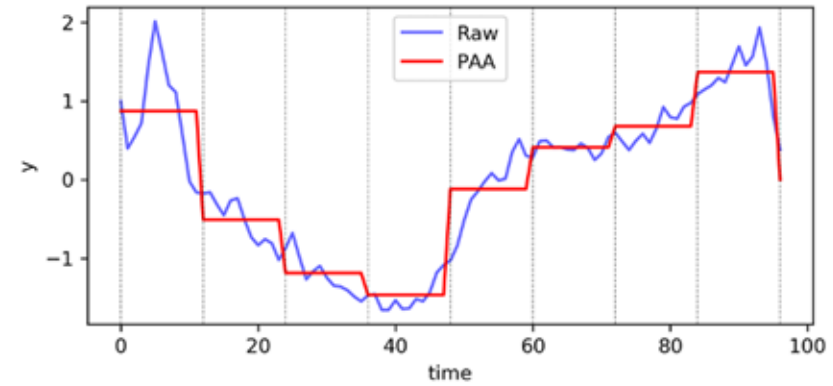
$$\bar{x}_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} x_j$$



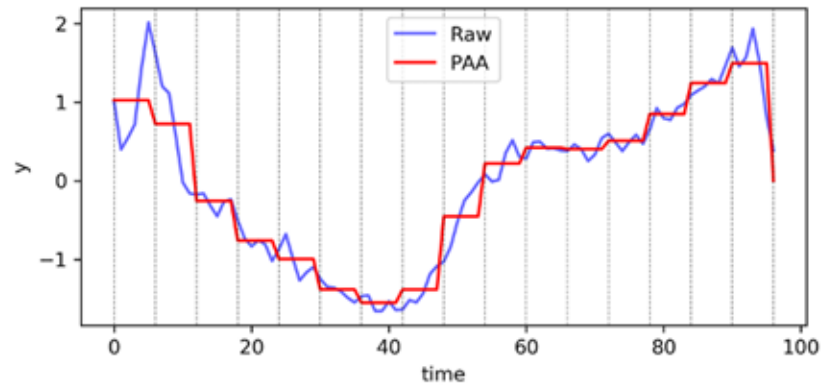
PAA - Example



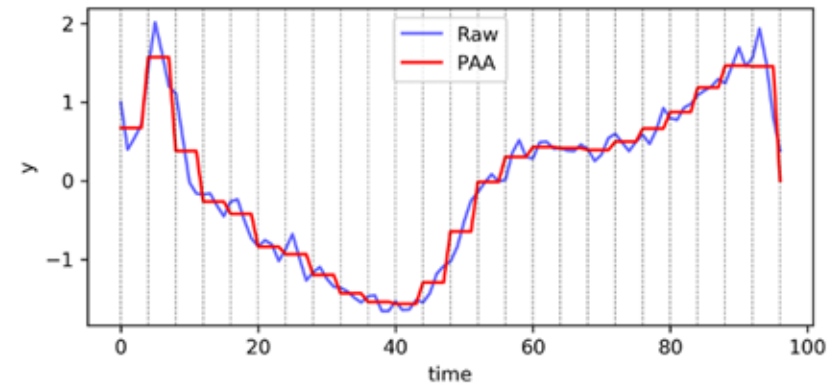
(a) segment $w = 2$.



(b) segment $w = 8$.

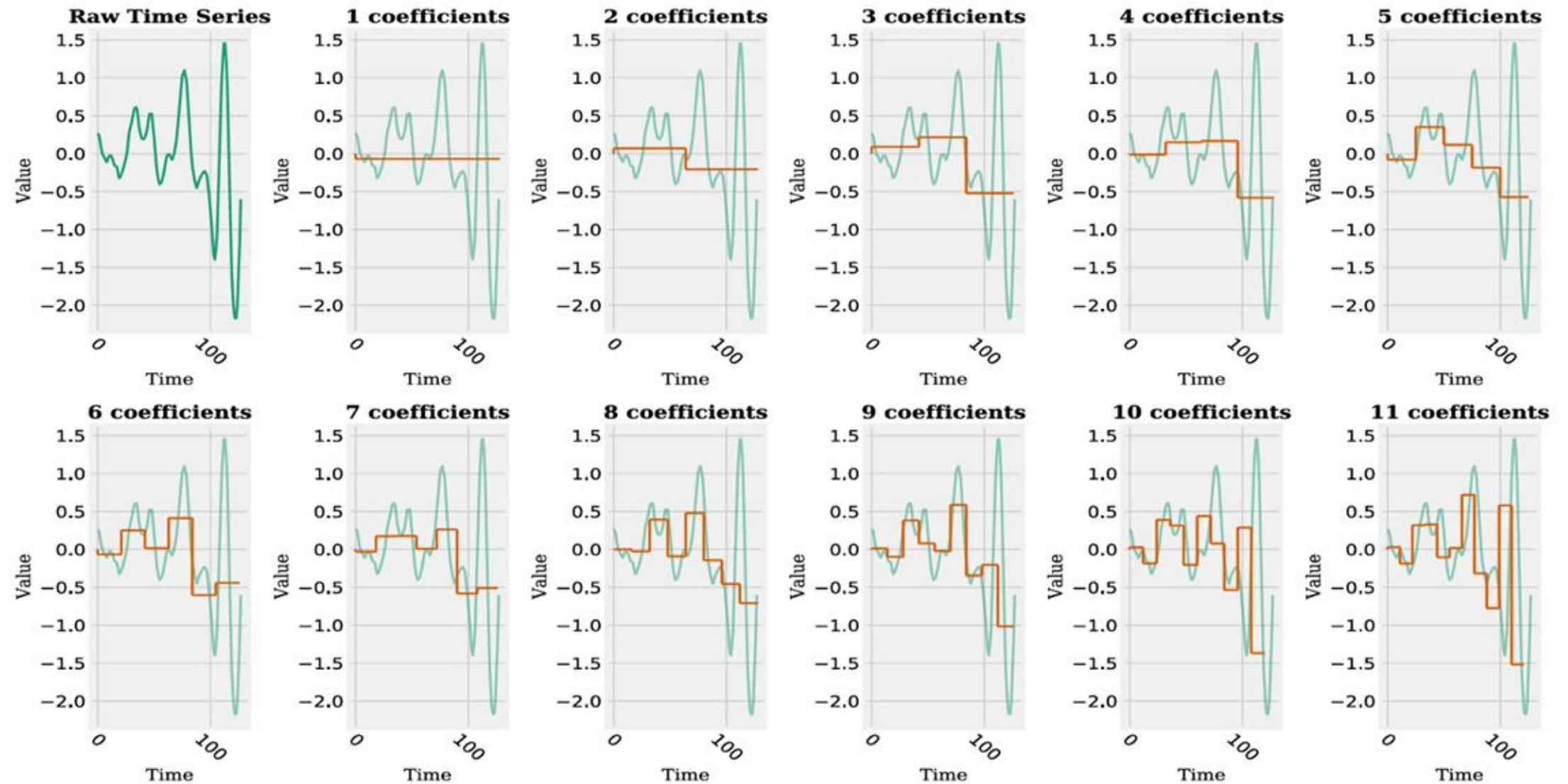


(c) segment $w = 16$.



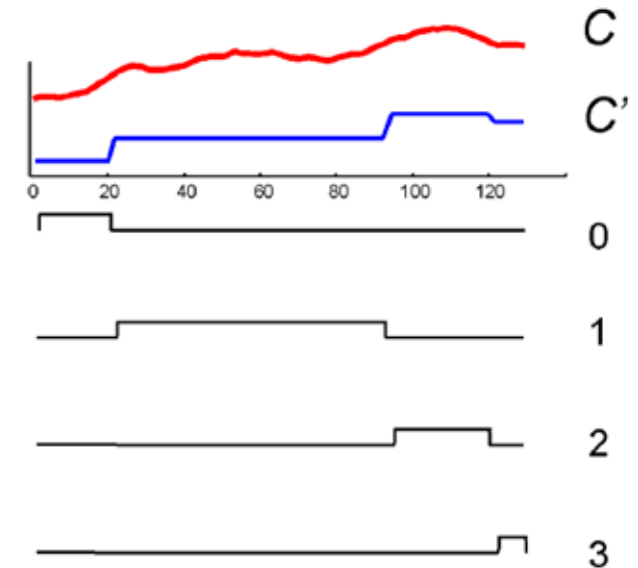
(d) segment $w = 24$.

PAA - Example



Adaptive Piecewise Constant Approximation (APCA)

- It allows the **segments to have arbitrary lengths**, which in turn needs two numbers per segment.
- The first number records the mean value of all the data points in segment, and the second number records the length of the segment.
- APCA has the advantage of being able to place a single segment in an area of low activity and many segments in areas of high activity.
- In addition, one has to consider the structure of the data in question.
- Pros:
 - Fast to calculate $O(n)$
 - Supports non Euclidean measures
 - Supports weighted Euclidean distance



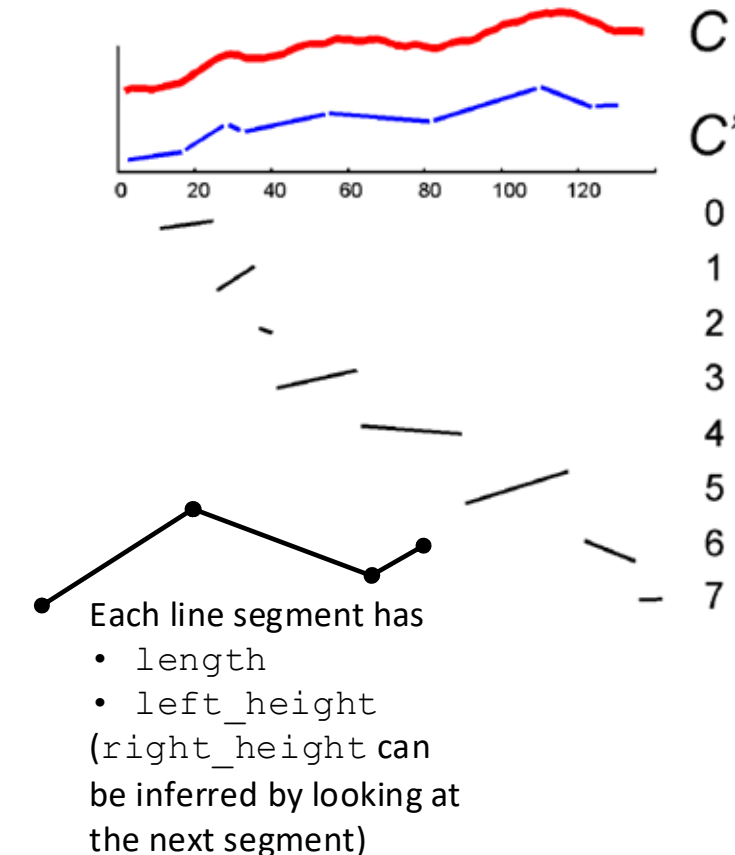
Piecewise Linear Approximation (PLA)



Karl Friedrich Gauss

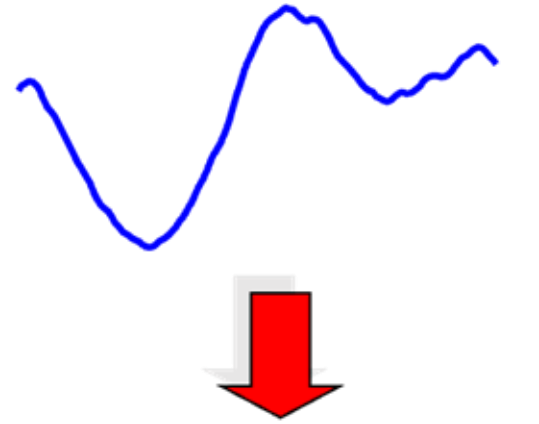
1777 - 1855

- Represent the time series as a sequence of straight lines.
- Lines could be connected or disconnected
- In the literature there are numerous algorithms available for **segmenting** time series.
- An open question is how to best choose K , the “optimal” number of segments used to represent a particular time series.
- This problem involves a tradeoff between accuracy and compactness, and clearly has no general solution.
- Pros:
 - data compression
 - noise filtering
 - able to support some interesting non-Euclidean similarity measures



Symbolic Aggregate Approximation (SAX)

- SAX converts a TS into a discrete format using a small alphabet size such that every part of the representation contributes about the same amount of information about the shape of the TS.
- First converts the time series to PAA representation, then convert the PAA to symbols.

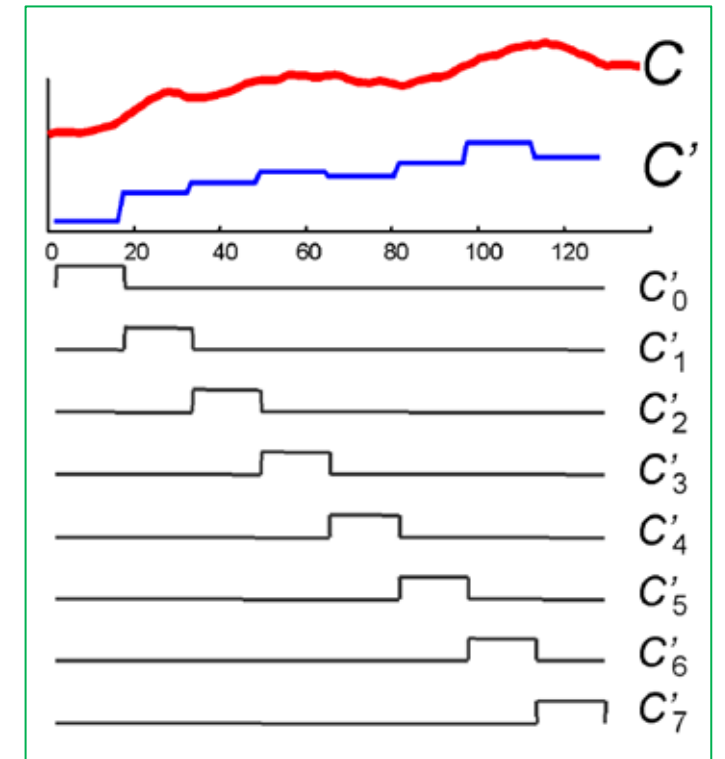


baabccbc



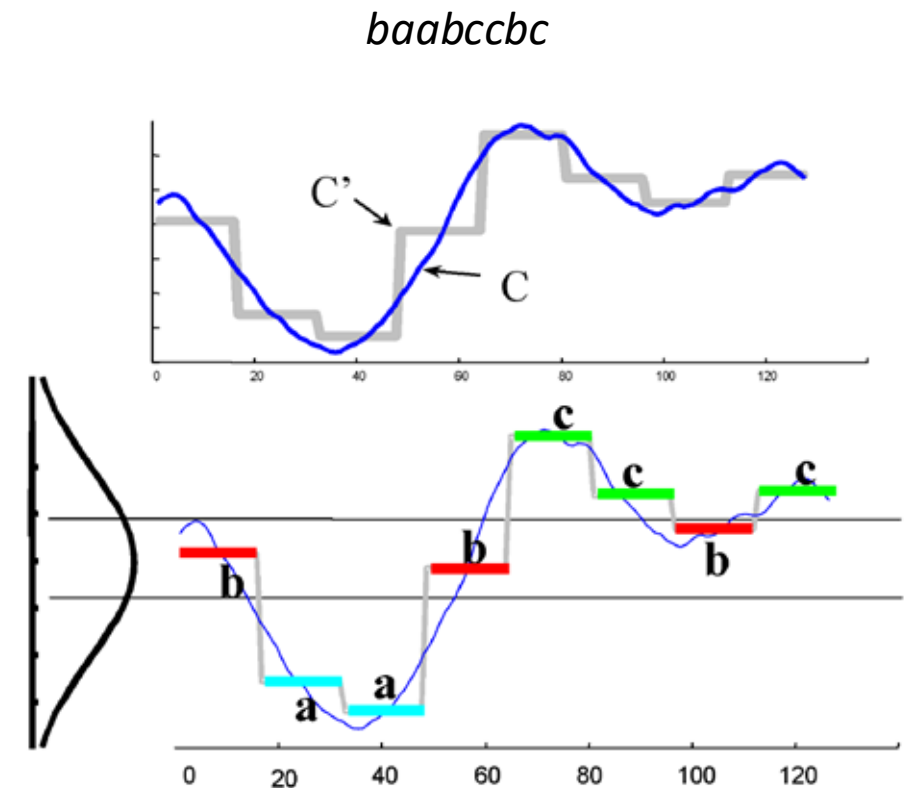
Symbolic Aggregate Approximation (SAX)

- Convert the data into a discrete format, with a **small alphabet size**.
- A time series T of length n is divided into w equal-sized segments; the values in each segment are then approximated and replaced by a **single coefficient**, which is their average.
- **Aggregating these w coefficients form the PAA representation of T .**
- Next, we **determine the breakpoints** that divide the distribution space into a equiprobable regions, where a is the alphabet size specified by the user
- The breakpoints are determined such that the probability of a segment falling into any of the regions is approximately the same.
- If the symbols are not equi-probable, some of the substrings would be more probable than others. Consequently, we would inject a probabilistic bias in the process.



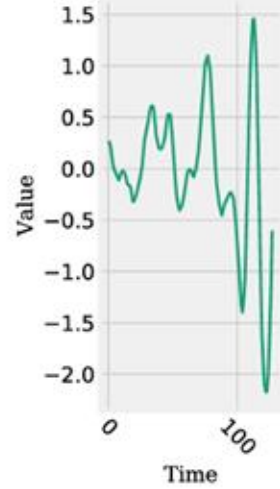
Symbolic Aggregate Approximation (SAX)

- Once the breakpoints are determined, each region is assigned a symbol.
- The PAA coefficients can then be easily mapped to the symbols corresponding to the regions in which they reside.
- The symbols are assigned in a bottom-up fashion, i.e., the PAA coefficient that falls in the lowest region is converted to “a”, in the one above to “b”, and so forth.

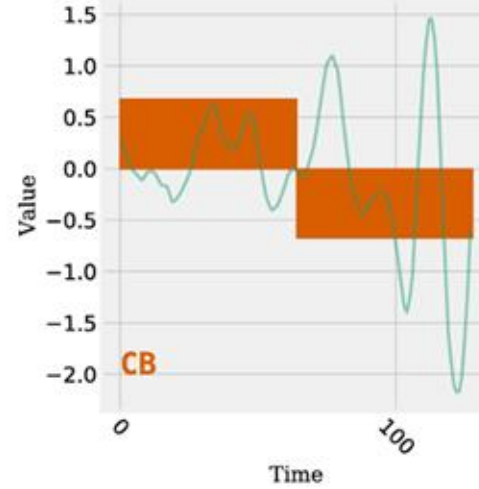


SAX - Example

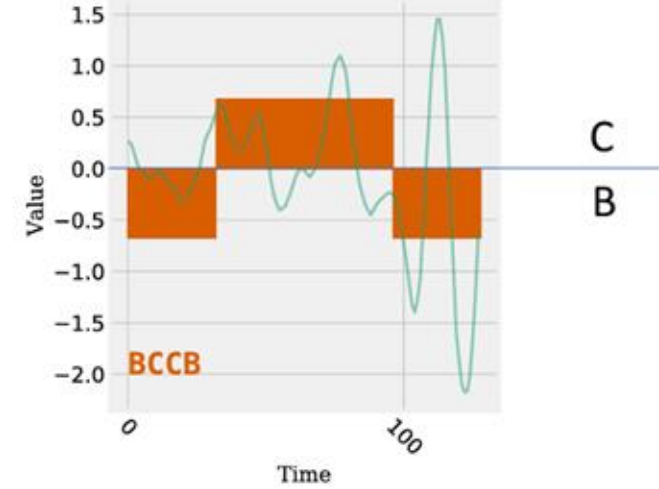
Raw Time Series



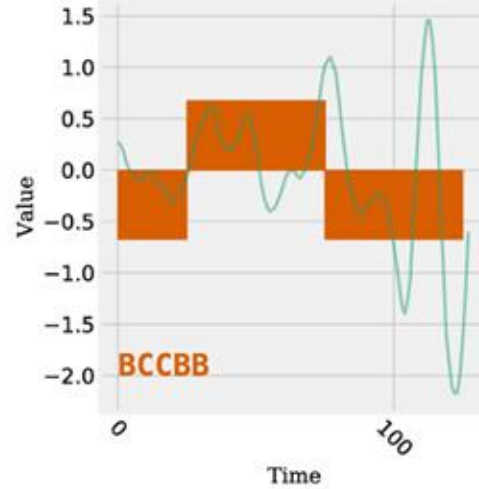
2 coefficients



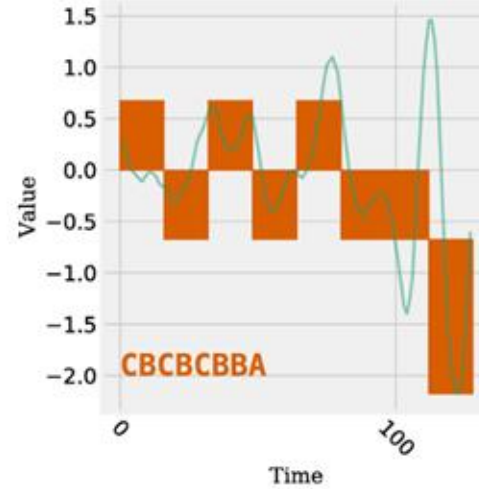
4 coefficients



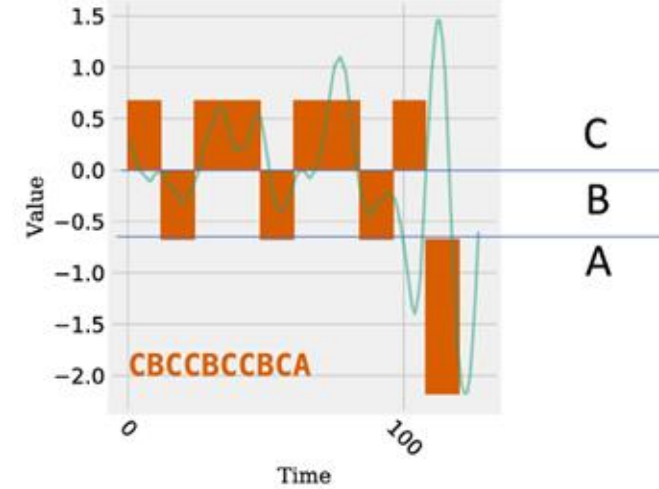
5 coefficients



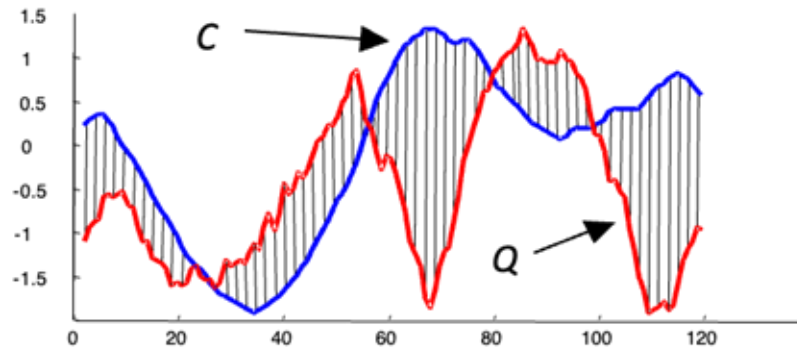
8 coefficients



10 coefficients



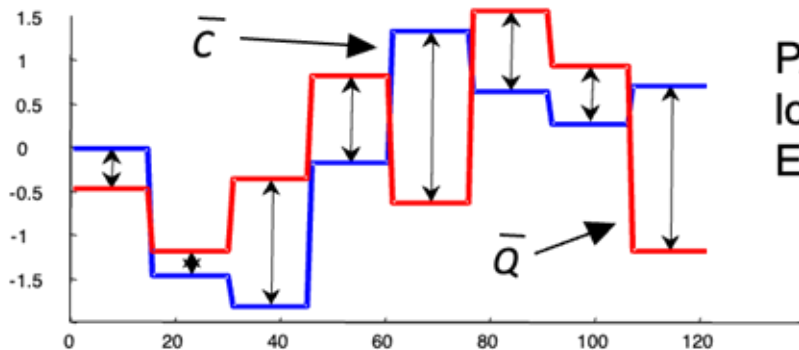
Distances and Approximations



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

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

$$\begin{aligned} \hat{C} &= \text{baabccbc} \\ \hat{Q} &= \text{babcacca} \end{aligned}$$

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

dist() can be implemented
using a table lookup or by
using for each symbol the
mean value of its region.

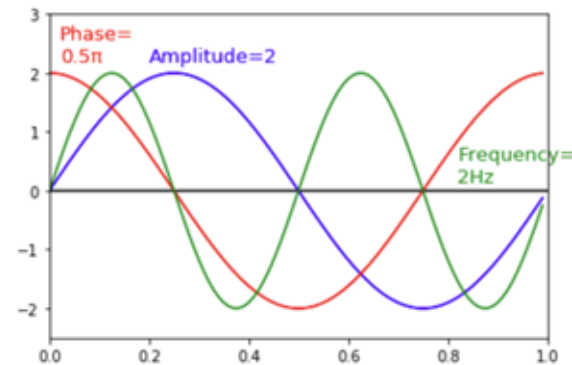
Discrete Fourier Transform (DFT)

- Apply a spectral decomposition of a signal
- DTF is a method to decompose functions depending on **time** into functions depending on **frequency**
- TS is a function depending on time
 - we have a value for temperature for each point in time.



Jean Fourier: 1768-1830

Frequency is
the number of
complete cycles



- DFT extracts **different** seasonality patterns from a single time series variable
- Example: Given an hourly temperature data set, DFT can detect the presence of **day/night variations** and **summer/winter variations**
 - it will tell you that those two seasonality (frequencies) are present in your data.

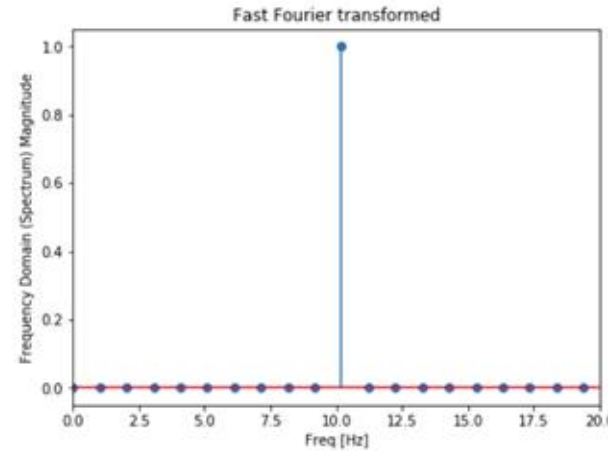
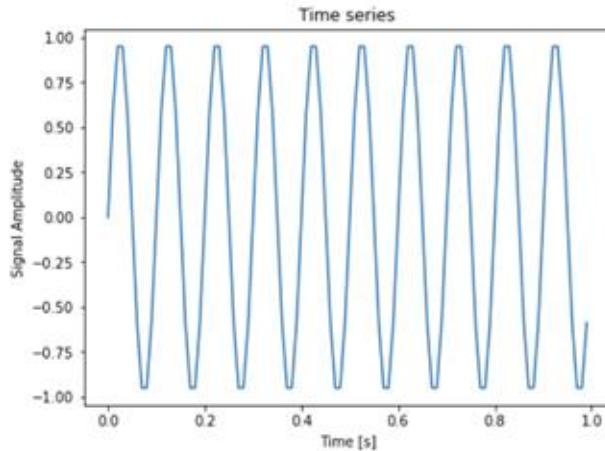
TS: a combination of seasonality, trend, and noise



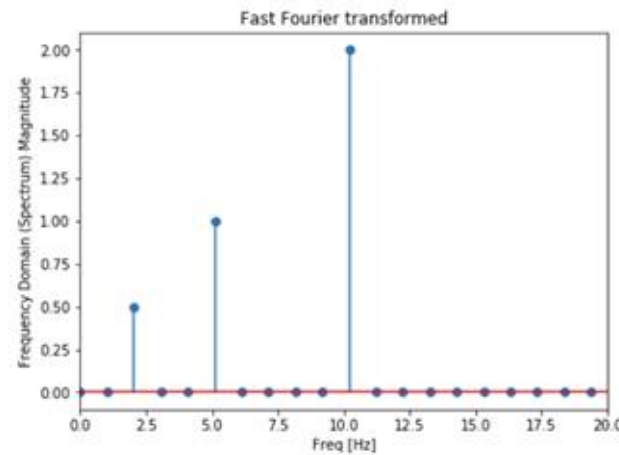
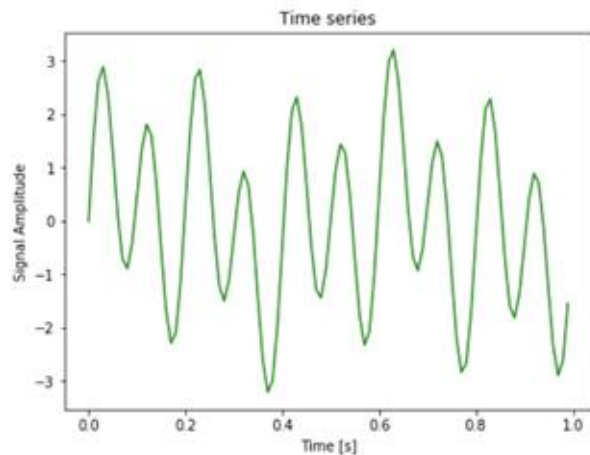
Discrete Fourier Transform (DFT)



Jean Fourier: 1768-1830

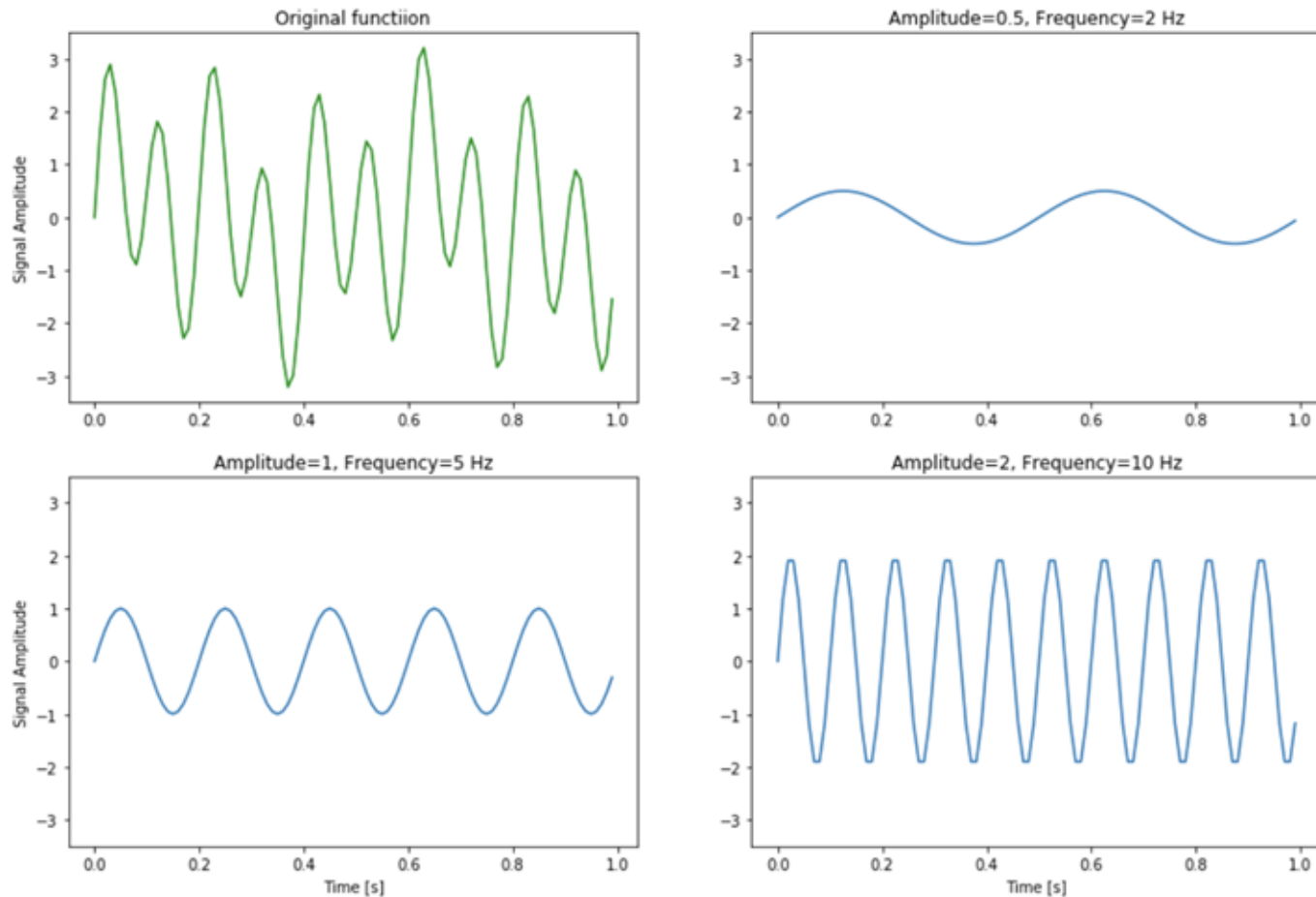


- A peak value at 10 Hz with a magnitude of one while all other frequencies are around zero.



- The original TS where has 10 complete cycles in a second with an amplitude of one.
- Data comprises of 3 different elementary components with 3 different frequencies (2, 5 and 10 Hz) at 3 different amplitudes (0.5, 1 and 2).

Discrete Fourier Transform (DFT)



- Data comprises of 3 different elementary components with 3 different frequencies (2, 5 and 10 Hz) at 3 different amplitudes (0.5, 1 and 2).
- Sine functions of the different components.

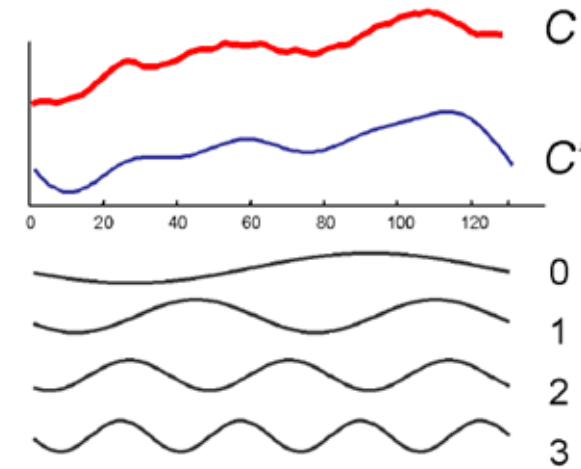
Discrete Fourier Transform (DFT)

- The basic idea of spectral decomposition is that any signal can be represented by the super position of a finite number of sine/cosine waves
- Each wave is represented by a single complex number known as a Fourier coefficient as a linear combination of sines and cosines
- Keep only the first $n/2$ coefficients
- Many of the Fourier coefficients have very low amplitude and thus contribute little to reconstructed signal.
- These low amplitude coefficients can be discarded without much loss of information thereby saving storage space.
- Pros
 - Good ability to compress most natural signals.
 - Fast, off the shelf DFT algorithms exist. $O(n\log(n))$.
- Cons
 - Difficult to deal with sequences of different lengths.
 - Cannot support weighted distance measures.



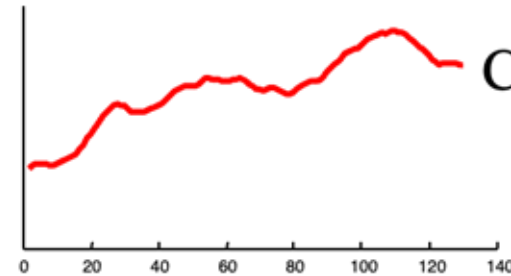
Jean Fourier

1768-1830

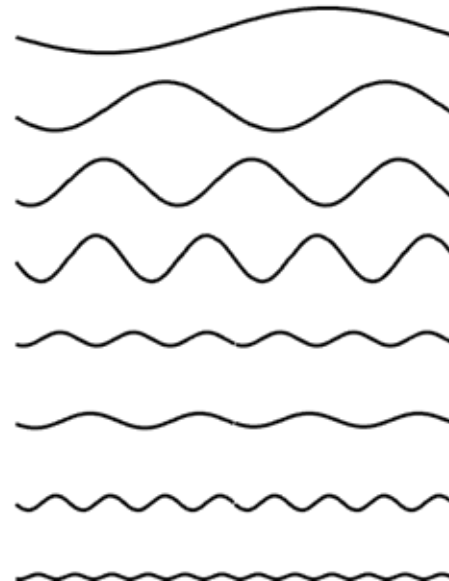


An Example of DFT

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



$n = 128$



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

An Example of DFT

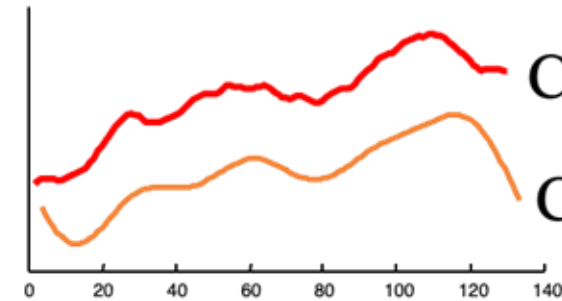
- ... however, note that the first few sine waves tend to be the largest (equivalently, the magnitude of the Fourier coefficients tend to decrease as you move down the column).
- We can therefore truncate most of the small coefficients with little effect.

$$n = 128$$

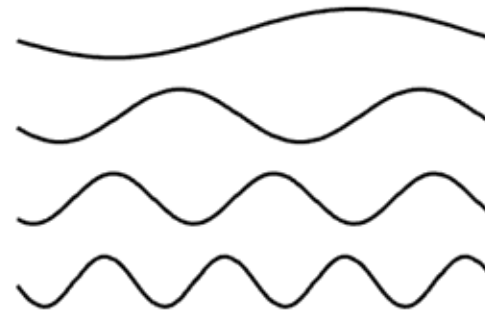
$$N = 8$$

$$C_{\text{ratio}} = 1/16$$

We have
discarded $\frac{15}{16}$
of the data.



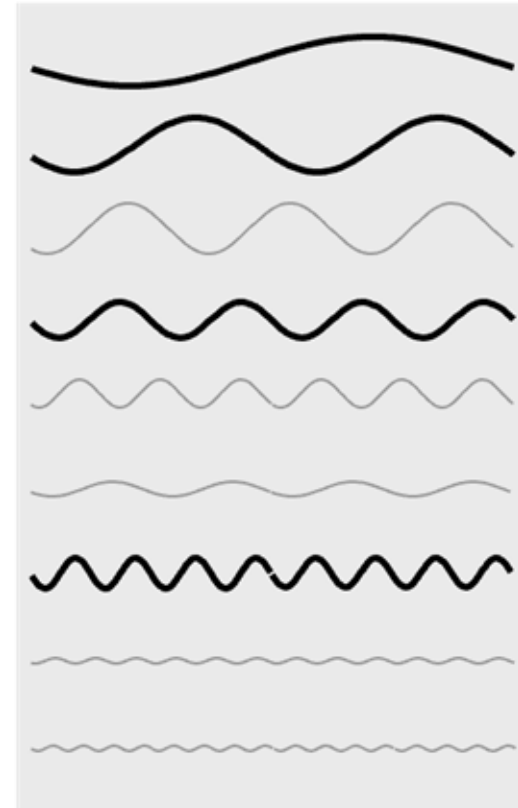
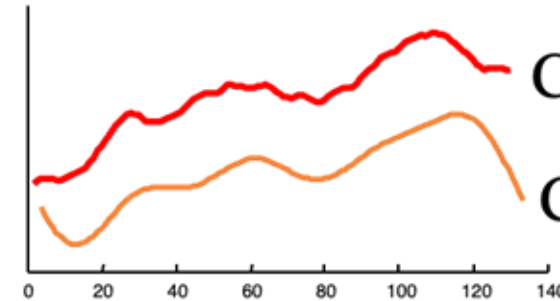
$n = 128$



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

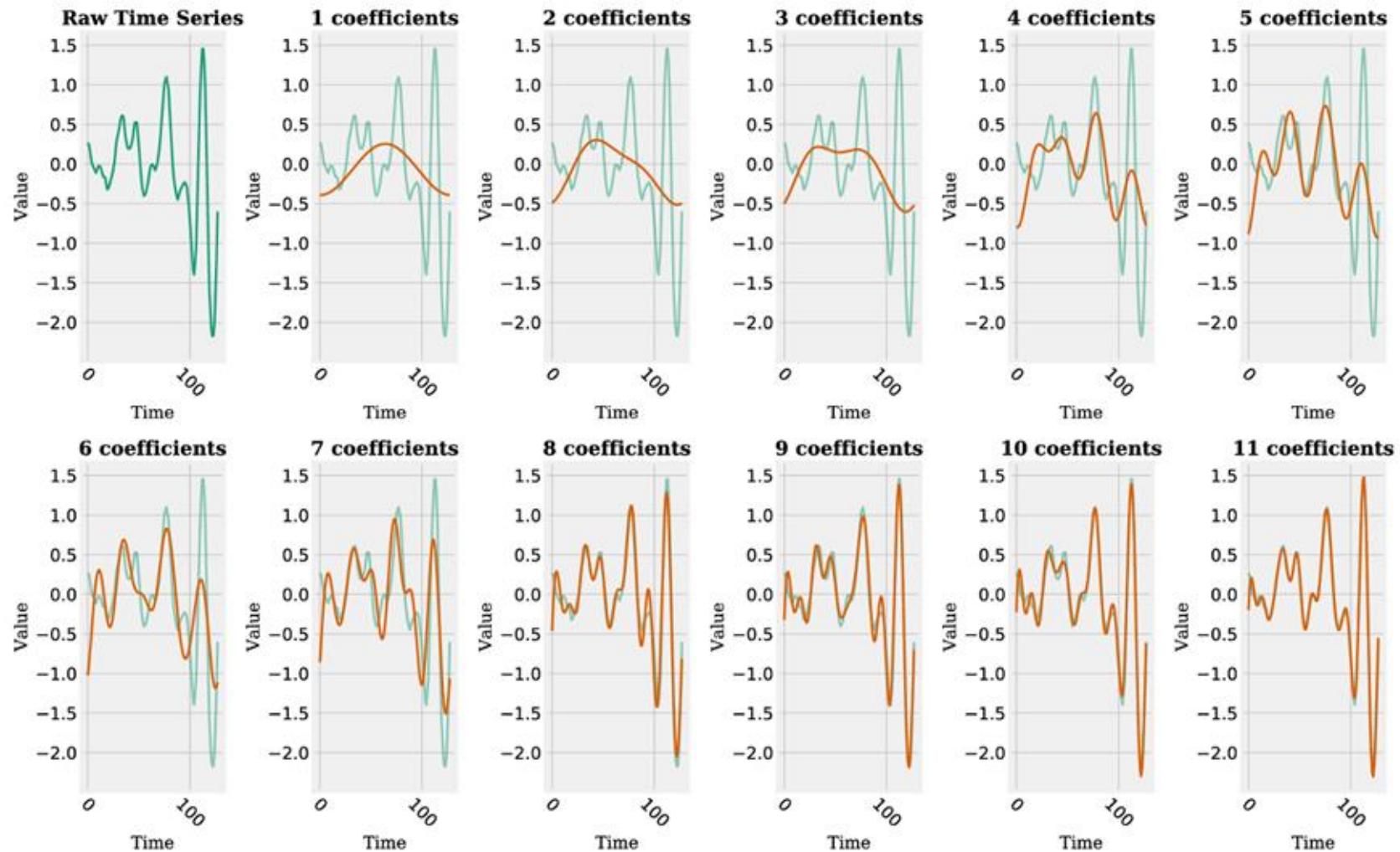
An Example of DFT

- Instead of taking the first few coefficients, we could take the best coefficients
- This can help greatly in terms of approximation quality but makes indexing hard.
- Note this applies also to Wavelets



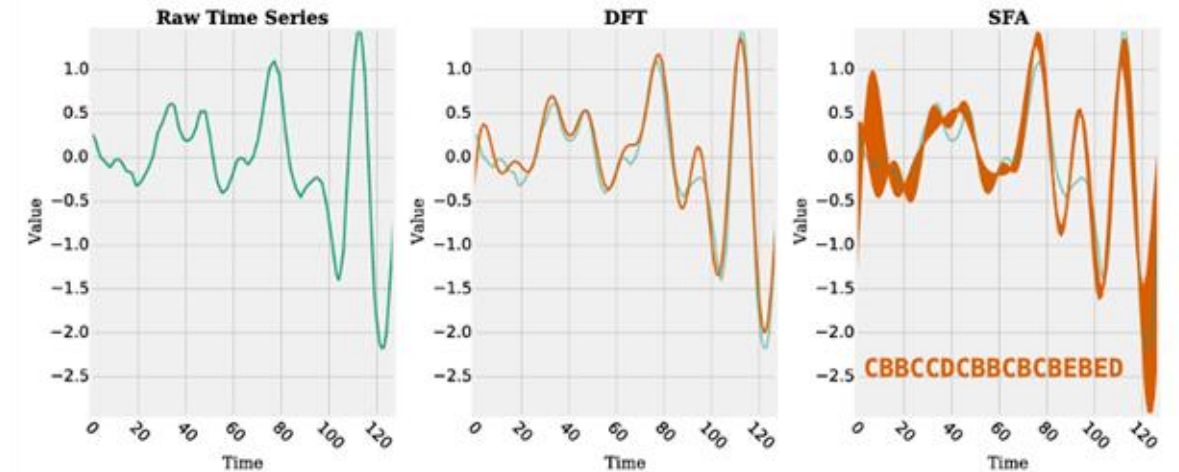
Truncated Fourier Coefficients	Fourier Coefficients	Raw Data
	1.5698	0.4995
<u>1.5698</u>	<u>1.0485</u>	0.5264
<u>1.0485</u>	0.7160	0.5523
<u>0.7160</u>	<u>0.8406</u>	0.5761
<u>0.8406</u>	0.3709	0.5973
<u>0.3709</u>	<u>0.4670</u>	0.6153
<u>0.4670</u>	0.2667	0.6301
<u>0.2667</u>	<u>0.1928</u>	0.6420
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

DFT - Example



Symbolic Fourier Approximation (SFA)

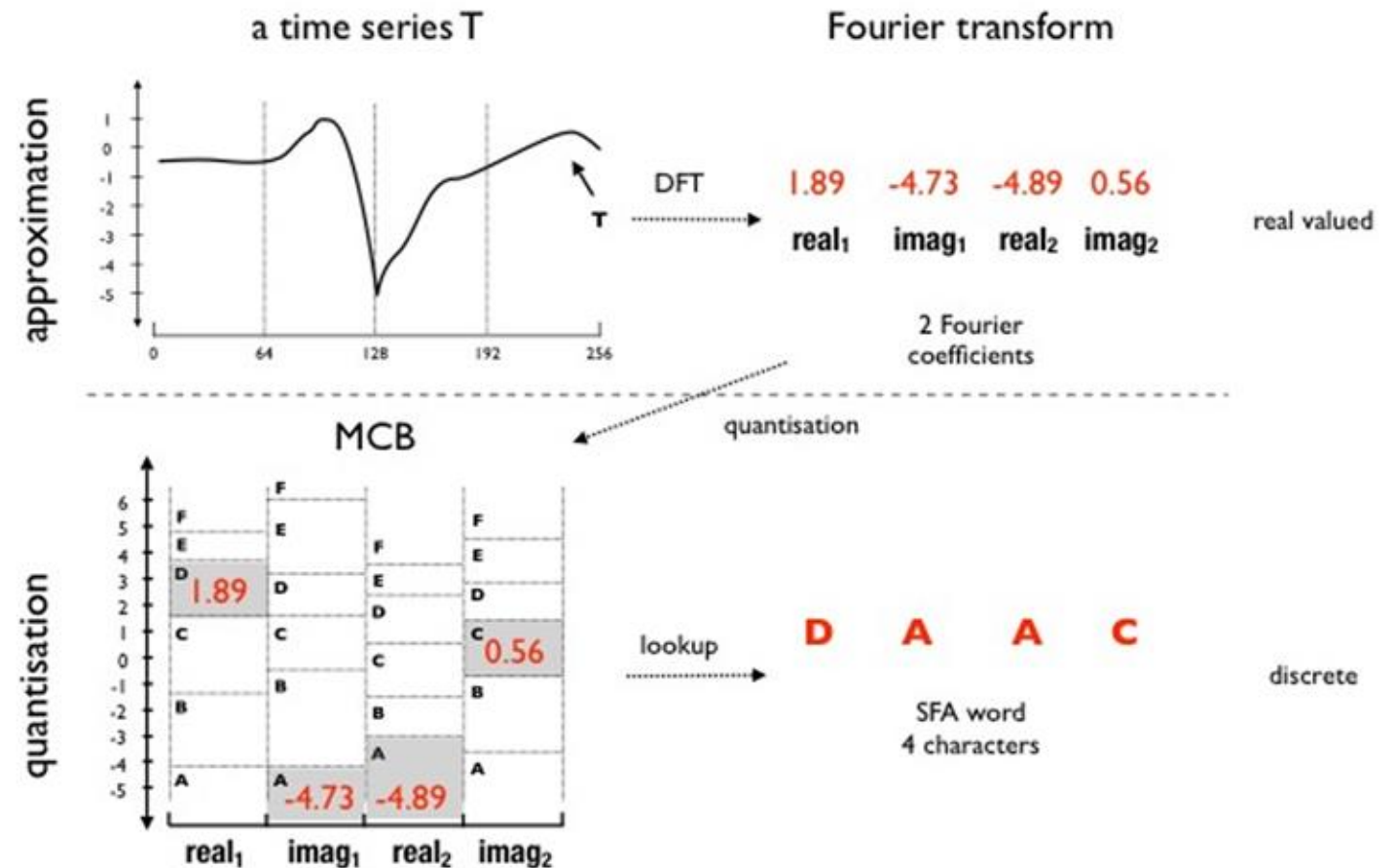
- SFA represents a TS with a word
- SFA is composed of
 - a) approximation using the Fourier transform
 - b) a data adaptive discretization
- The discretization intervals are learned from the Fourier transformed data distribution rather than using fixed intervals with Multiple Coefficient Binning (MCB)



Raw:	DFT	Discretization
0.2679	0	C
0.2480	-8.81	B
0.1828	-20.7	B
0.0817	-11.9	C
0.0051	-6.28	C
-0.023	-8.02	D
-0.052	-0.67	C
-0.082	15.31	B
-0.111	-18.7	B
-0.075	-18.36	C
-0.032	-5.67	B
-0.022	-16.84	C
-0.029	-8.919	B
[...]	[...]	[...]

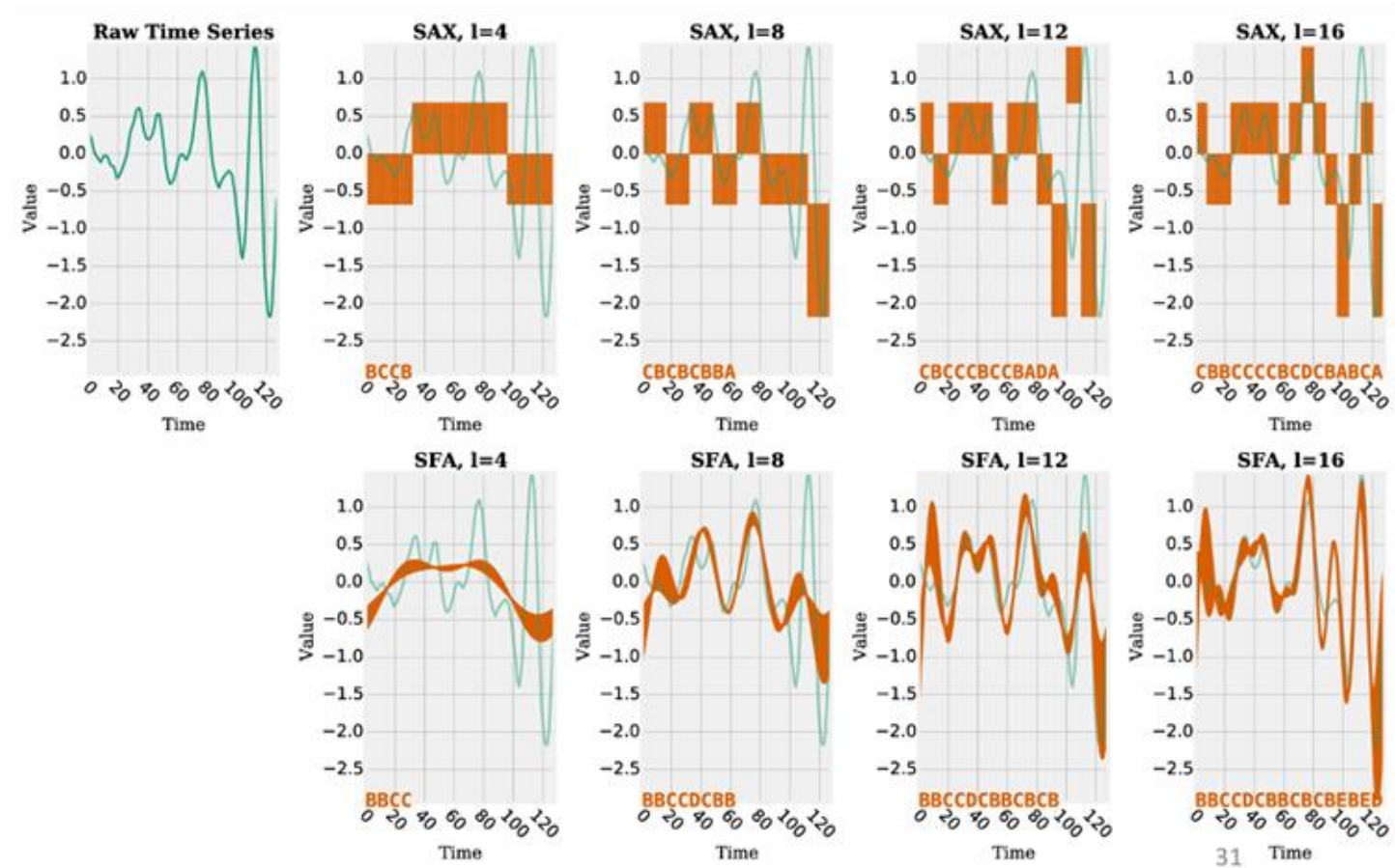
SFA Discretization

- The objective of MCB is to minimize the loss of information introduced by discretization.
- This is achieved by applying a discretization for each coefficient.
- MCB uses different breakpoints for each symbol on each coefficient.



Comparison of SFA and SAX

- Roughly we can say
 - $\text{SAX} = \text{PAA} + \text{Discretization}$
 - $\text{SFA} = \text{DFT} + \text{Discretization}$
- Approximation cause a loss of information
- Discretization augments the level of information loss.
- The higher the number of symbols and the alphabet size, the more exact is the representation.



Properties of Symbolic Representations

- Noise removal: discretization
- String representations: allows for string domain algorithms like hashing or the bag-of-words to be applied
- Dimensionality reduction: allow for indexing high dimensional data
- Storage reduction: sequences have a much lower memory footprint than real-valued time series

Singular Value Decomposition (SVD)

- SVD is a factorization method that decomposes a matrix into three other matrices: U , S , and V^T (transpose of V):

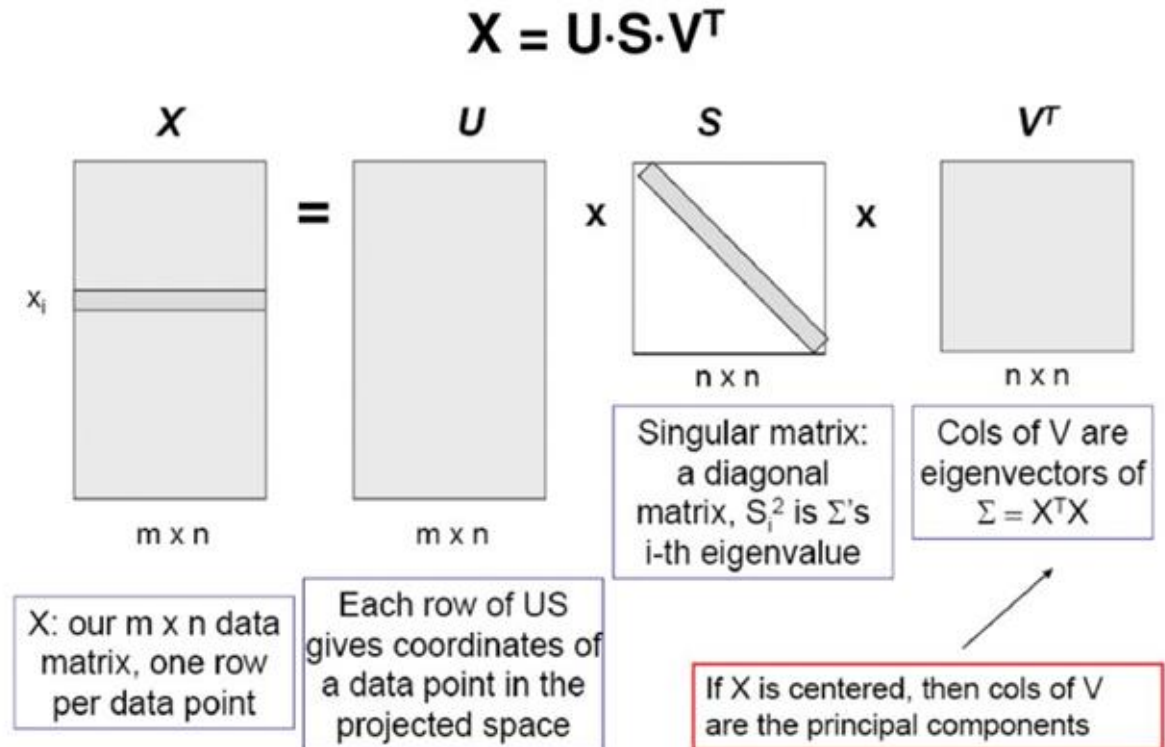
- $X = U S V^T$

- Input matrix X ($m \times n$)

- U ($m \times m$) contains orthogonal columns that represent the left singular vectors.

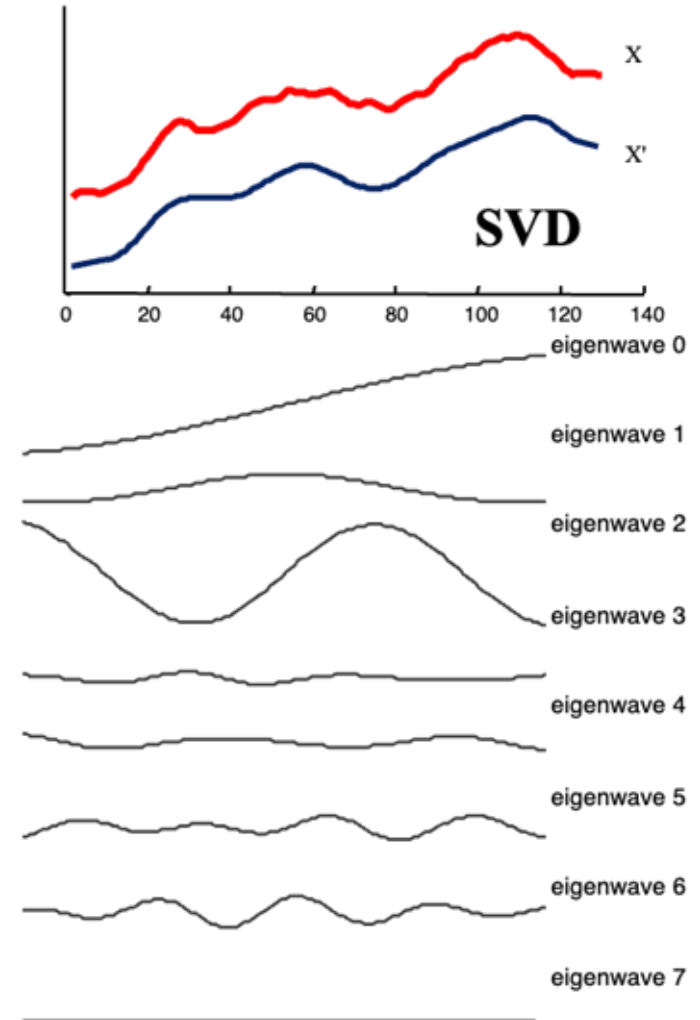
- S ($m \times n$) is a diagonal matrix containing the singular values.

- V^T ($n \times n$) contains orthogonal rows representing the right singular vectors.



SVD for Time Series Approximations

- SVD is similar to DFT and SFT in that it represents the shape in terms of a linear combination of basis shapes.
- DFT and SFT are **individual approximations** as they examine one TS at a time. They are completely independent of the rest of the data.
- SVD is a **global approximations** as the entire dataset is examined and is then rotated such that the first axis has the maximum possible variance, the second axis has the maximum possible variance orthogonal to the first, the third axis has the maximum possible variance orthogonal to the first two, etc.
- The global nature of SVD is both a weakness and a strength.



Principal Component Analysis (PCA)

•PCA is a statistical procedure that aims to transform data into a new coordinate system where the axes are the principal components. These components are orthogonal and capture the maximum variance in the data.

- 1.**Standardize the Data:** Normalize the data X to have zero mean and unit variance matrix C .
- 2.**Calculate Covariance Matrix:** Calculate the covariance matrix $\Sigma = C^T C$ of the standardized data.
- 3.**Compute Eigenvectors and Eigenvalues:** Compute the eigenvectors and eigenvalues of the covariance matrix Σ .
- 4.**Select Principal Components:** Sort eigenvalues in descending order and choose the top- k eigenvalues to form principal components.
- 5.**Transform Data:** Project the original data X into the principal components to create a lower-dimensional representation.

Relationships between SVD and PCA

Dealing with data

- PCA primarily deals with the covariance structure of the data.
- SVD does not rely on a covariance matrix. It is a factorization that decomposes the original data without computing covariance.

Computations

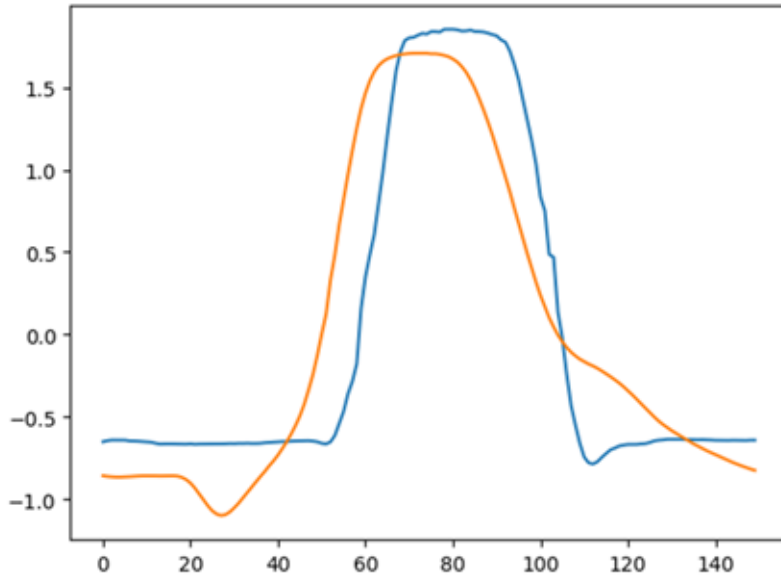
- Both PCA and SVD involve eigen-decomposition.
- PCA performs eigen-decomposition on the covariance matrix of the data which is a square symmetric matrix of size $m \times m$ where m is the number of time stamp.
- SVD performs eigen-decomposition on the data matrix itself of size $n \times m$ where n is the number of TS and m is the number of number of time stamps.

Relationships between SVD and PCA

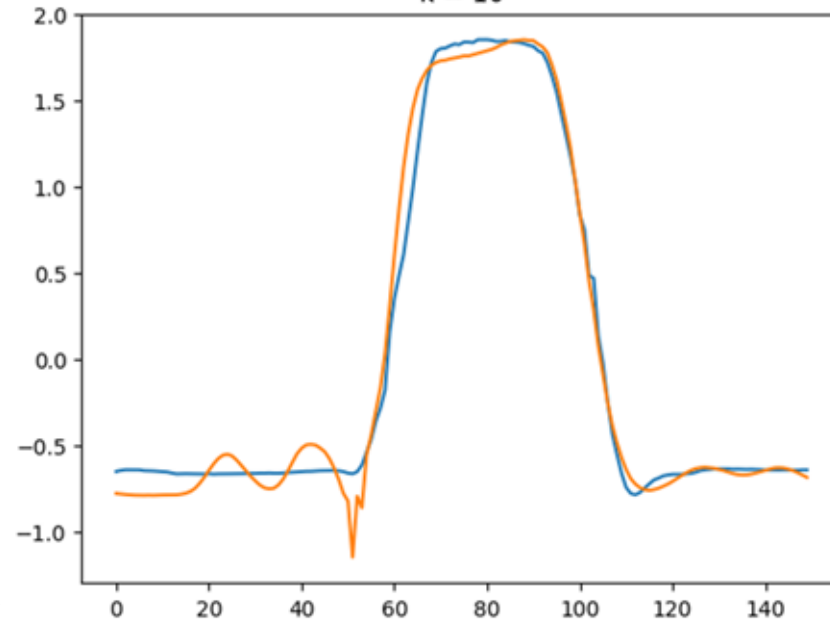
- **PCA is a specific application of SVD**, primarily used for dimensionality reduction, while SVD is a more general matrix decomposition technique with broader applications in linear algebra and data analysis.
- PCA can be solved using SVD
- PCA focuses on the covariance structure and tries to maximize variance along orthogonal axes
- SVD focuses on matrix factorization and can handle cases where data is missing.
- **From an application perspective they are used interchangeably.**

PCA - Example

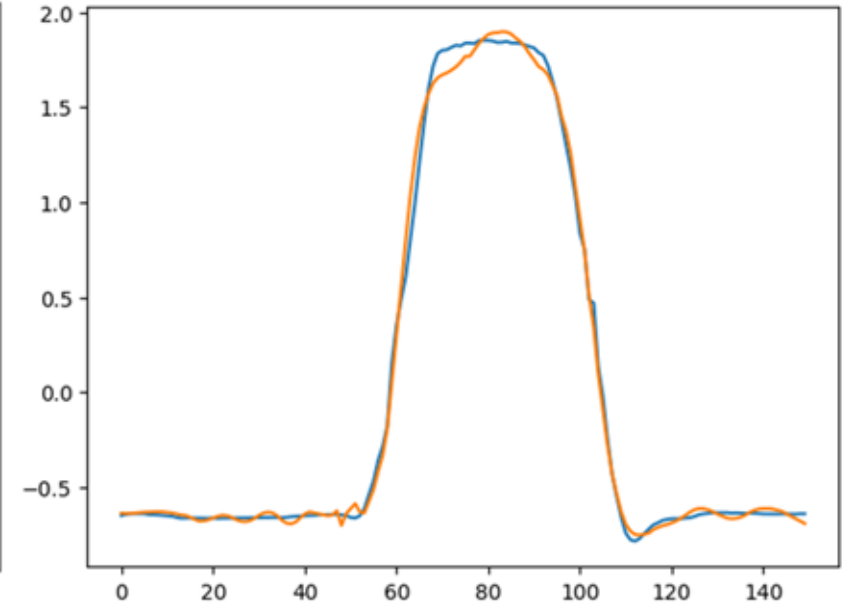
$k = 2$



$k = 10$



$k = 20$



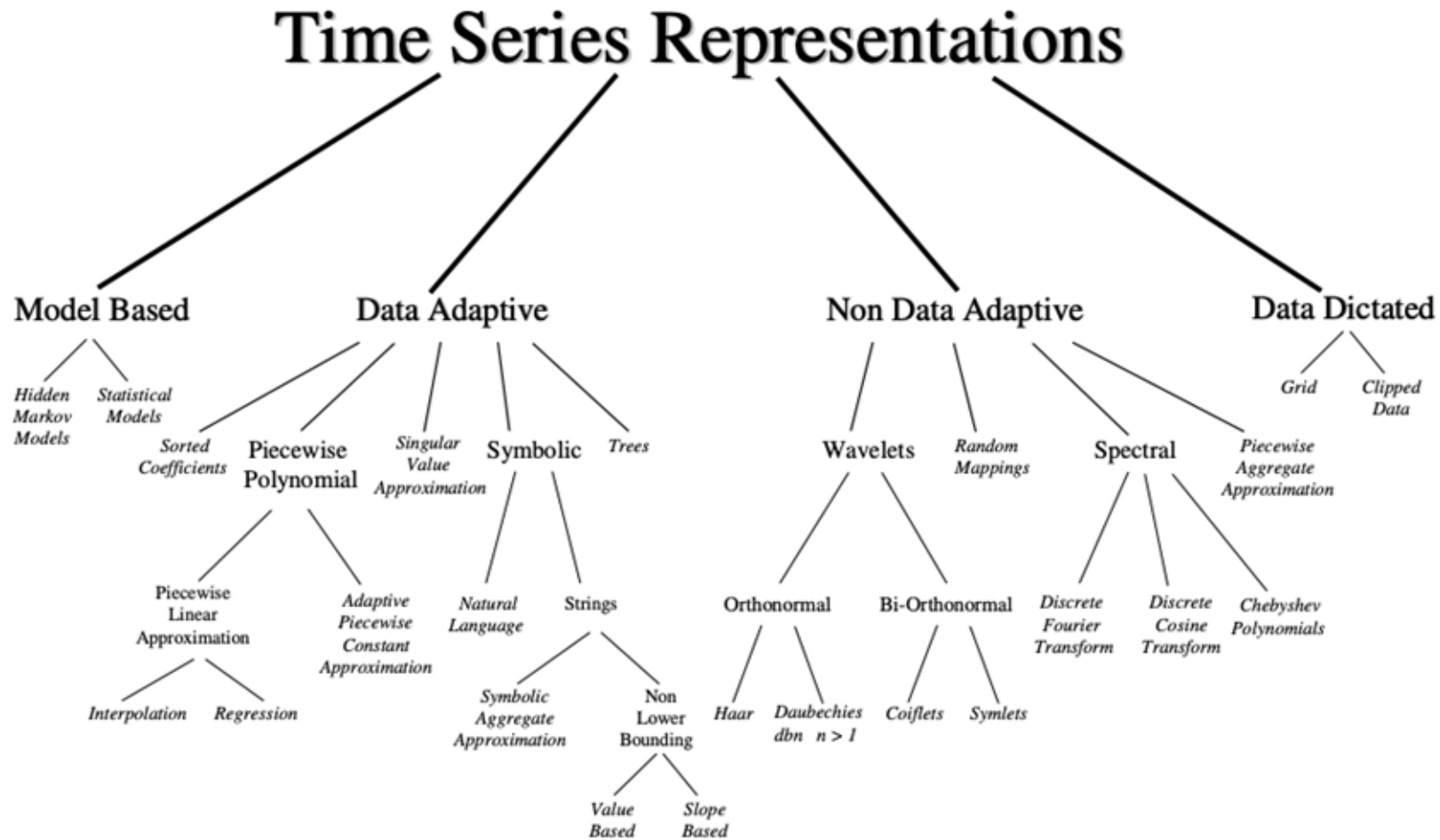
Remarks on Approximations

- None of the representations can be superior for all tasks.
- No research has proved how one should choose the best representation for the problem at hand and data of interest.
- The literature is not even consistent on nomenclature.
 - Example: Piecewise Aggregate Approximation
 - Piecewise Flat Approximation (Faloutsos et al., 1997)
 - Piecewise Constant Approximation
 - Segmented Means

Approximations, Distances and Normalizations

- It does not make any sense to use a distance function accounting for time like DTW if a Time-Independent approximation is used.
- Thus, do not use DTW after DTF, SFA, SVD, PCA.
- It does make sense to use DTW after Time-Dependent approximations.
- Thus, you can use DTW after PAA or SAX.
- Normalizations can be applied before and/or after approximations depending on the objective of your TSA task.
- Time series normalizations does not make any sense after a Time-Independent approximation.
- In that case traditional “column-wise” normalizations like Min-Max scaling or Z-Score normalization should be used.

Summary of Time Series Representation



Clustering

Clustering Time Series

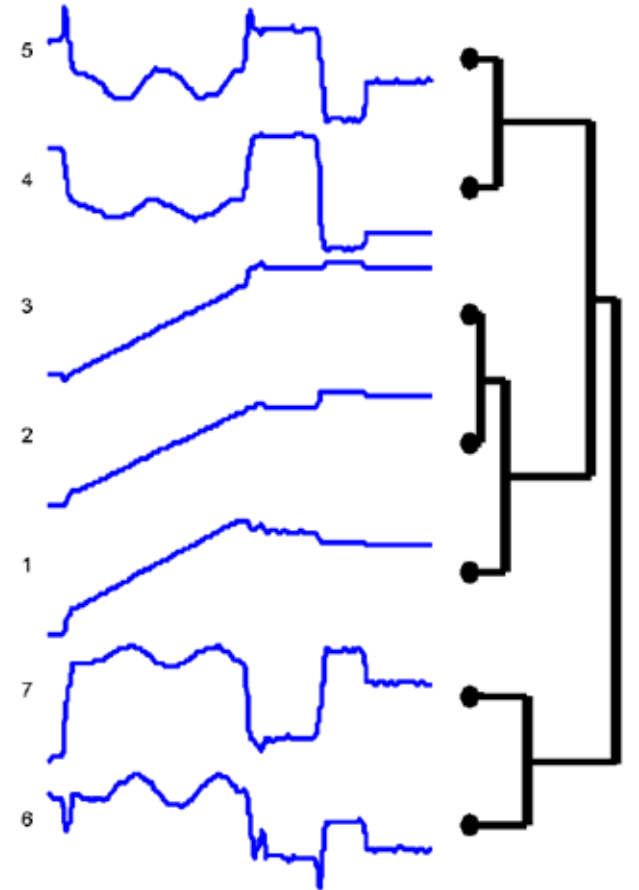
- It is based on the similarity between time series.
- The most similar data are grouped into clusters, but the clusters themselves should be dissimilar.
- These groups to find are not predefined, i.e., it is an unsupervised learning task.
- The two general methods of time series clustering are
 - Partitional Clustering and
 - Hierarchical Clustering

Types of Time Series Clustering

- ***Whole clustering***: similar to that of conventional clustering of discrete objects. Given a set of individual time series data, the objective is to group similar time series into the same cluster.
- ***Features-based clustering***: extract features, or time series motifs (see next lectures) as the features and use them to cluster time series.
- ***Compression-based clustering***: compress time series and run clustering on the compressed versions.
- ***Subsequence clustering***: given a single time series, subsequence clustering is performed on each individual time series extracted from the long time series with a sliding window.

Hierarchical Clustering

- It ***computes pairwise distance***, and then merges similar clusters in a bottom-up fashion, without the need of providing the number of clusters
- It is one of the best tools to data evaluation, by creating a dendrogram of several time series from the domain of interest.
- Its application is limited to small datasets due to its quadratic computational complexity.



Partitional Clustering

- Typically uses the K-Means algorithm (or some variant) to optimize the objective function by minimizing the sum of squared intra-cluster errors.
- K-Means is perhaps the most commonly used clustering algorithm in the literature, one of its shortcomings is the fact that the number of clusters, K , must be pre-specified.
- Also the ***distance function plays a fundamental role*** both for the quality of the results and for the efficiency.

References

- Forecasting: Principles and Practic. Rob J Hyndman and George Athanasaopoulus. (<https://otexts.com/fpp2/>)
- Time Series Analysis and Its Applications. Robert H. Shumway and David S. Stoffer. 4th edition. (<http://www.stat.ucla.edu/~frederic/221/W21/tsa4.pdf>)
- Mining Time Series Data. Chotirat Ann Ratanamahatana et al. 2010. (https://www.researchgate.net/publication/227001229_Mining_Time_Series_Data)
- Dynamic Programming Algorithm Optimization for Spoken Word Recognition. Hiroaki Sakode et al. 1978.
- Experiencing SAX: a Novel Symbolic Representation of Time Series. Jessica Line et al. 2009
- Compression-based data mining of sequential data. Eamonn Keogh et al. 2007.

