

# Distances, Approximations and Global Structural Features

---

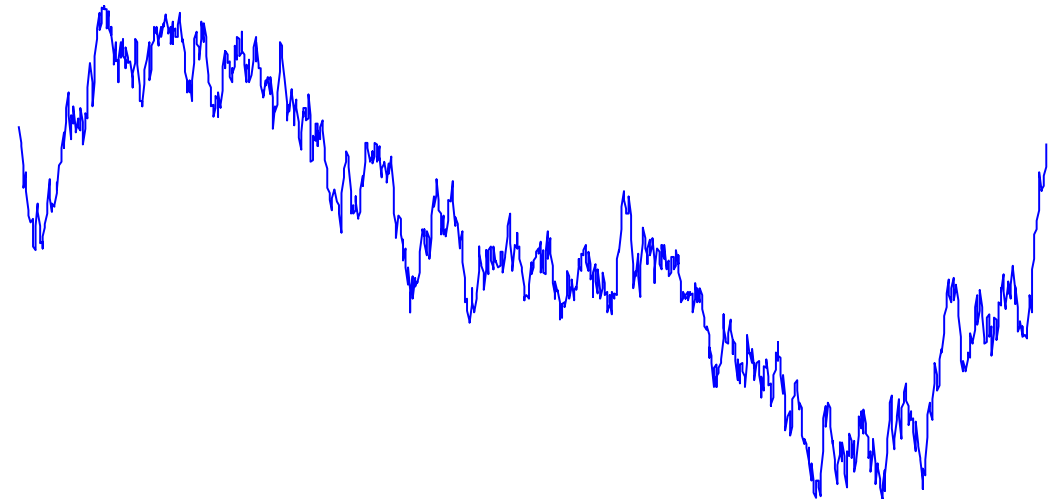
Francesco Spinnato, Riccardo Guidotti



# Syllabus

---

- Distances
  - Euclidean Distance
  - Dynamic Time Warping
- Approximations
  - Piecewise Aggregate Approximation
  - Symbolic Aggregate approXimation
  - Discrete Fourier Transformation
  - Symbolic Fourier Approximation
  - Singular Value Decomposition
  - Principal Component Analysis
- Global Structural Features
  - Simple Standard Features
  - Catch22 Features
  - TSPfresh Features

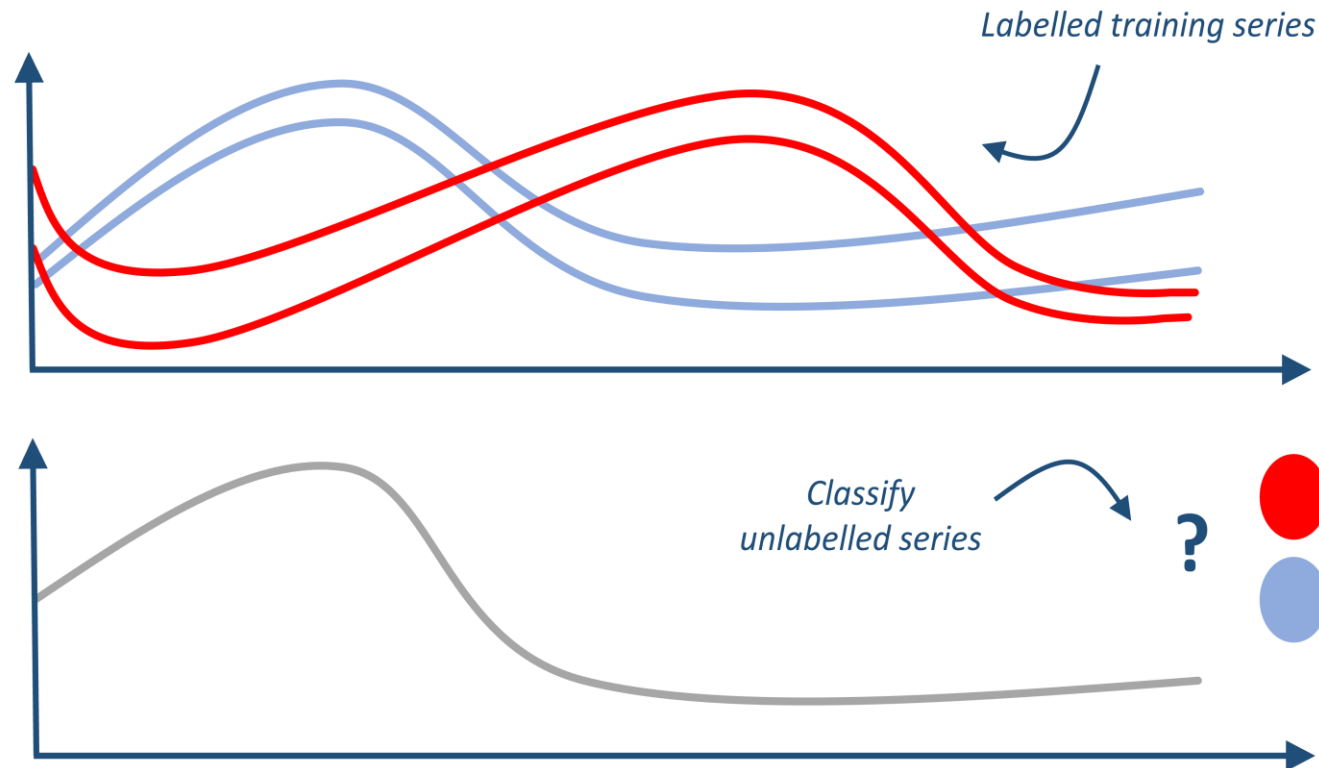


# A Simple ML Classifier

---

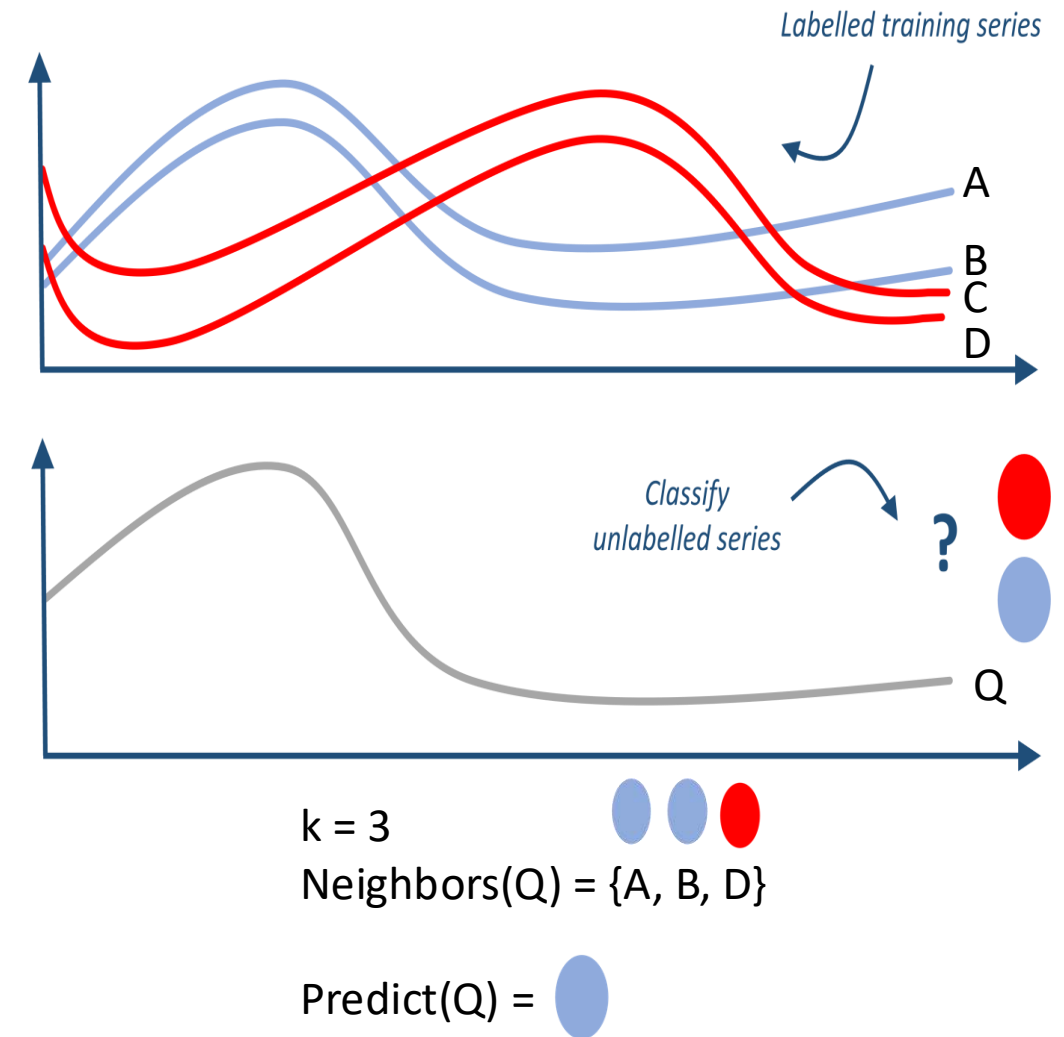
# Time Series Classification - TSC

- Given a dataset  $X = \{T_1, \dots, T_n\}$ , TSC is the task of training a model  $f$  to predict an exogenous categorical output  $y$  for each time series  $T$ , i.e.,  $f(T) = y$ .



# Nearest-Neighbor Classifier (K-NN)

- Basic idea: If it walks like a duck, quacks like a duck, then it is probably a duck.
- Given a set of training records, and a test record:
  1. Compute the distances from the test to the training records.
  2. Identify the  $k$  “nearest” records.
  3. Use class labels of nearest neighbors to determine the class label of test record (e.g., by taking majority vote).



# Performance Evaluation

---

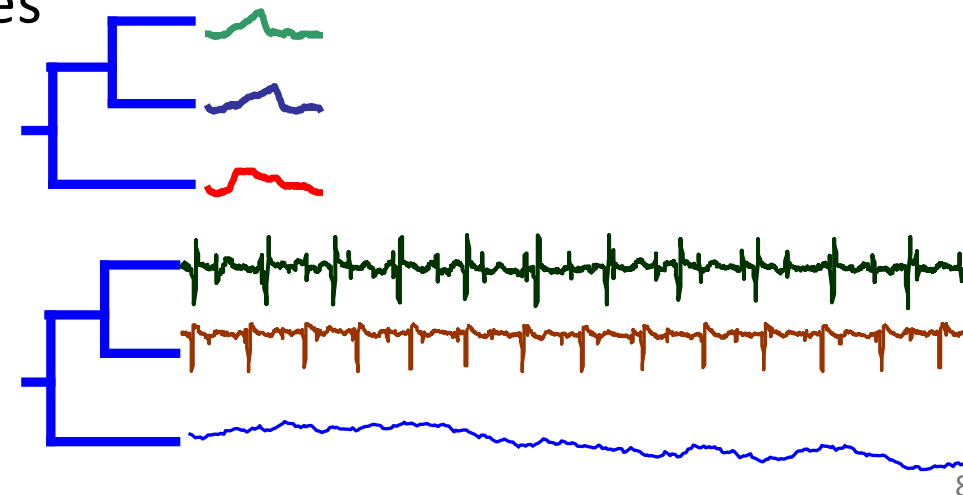
- Let suppose we have a vector  $y$  of actual/real class labels:
- $y = [0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0]$
- Let name  $y'$  the vector returned by a kNN:
- $y' = [0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0]$
- **Accuracy** = # record correctly classified / # records classified
- Accuracy =  $11 / 16 = 0.69$

# Time Series Distances

---

# Distances and Similarities

- Time series problems such as classification, forecasting, clustering, etc. require the usage of a notion of distance or similarity.
- What is similarity?
- It is the quality or state of being similar, likeness, resemblance, as a similarity of features.
- In TSA we recognize two types of similarity measures depending on the data representation considered:
  - **shape-based similarity**
  - **structural-based similarity**



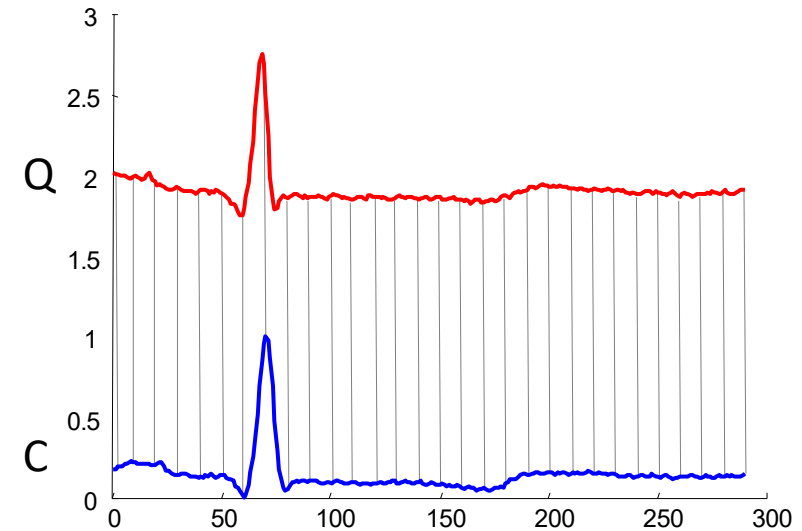
# Shape vs Structural Similarities

## Shape-based Similarity

- The original values of the time series are compared taking time into account.
- Better for short time series.

## Structural Similarity

- Time series are transformed into an alternative representation where the novel features are time-independent.
- Better for long time series.



|   | min | max | mean | std |
|---|-----|-----|------|-----|
| Q | 1.8 | 2.9 | 2.0  | 1.3 |
| C | 0.0 | 1.0 | 0.2  | 1.2 |

# Euclidean Distance

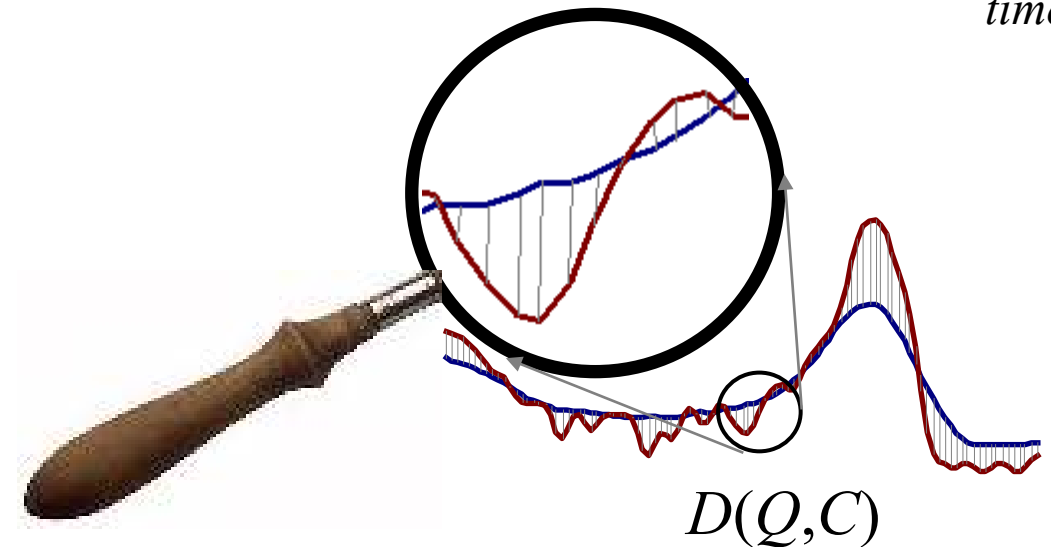
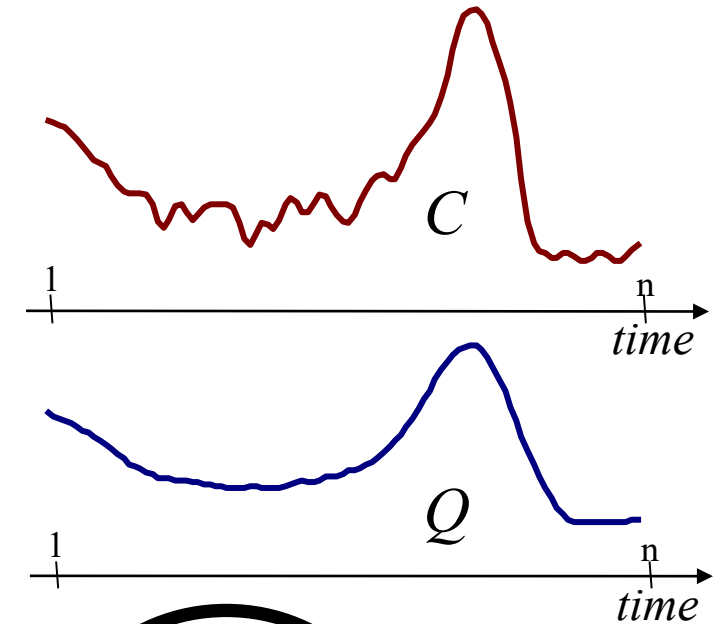
- Given two time series:

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

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

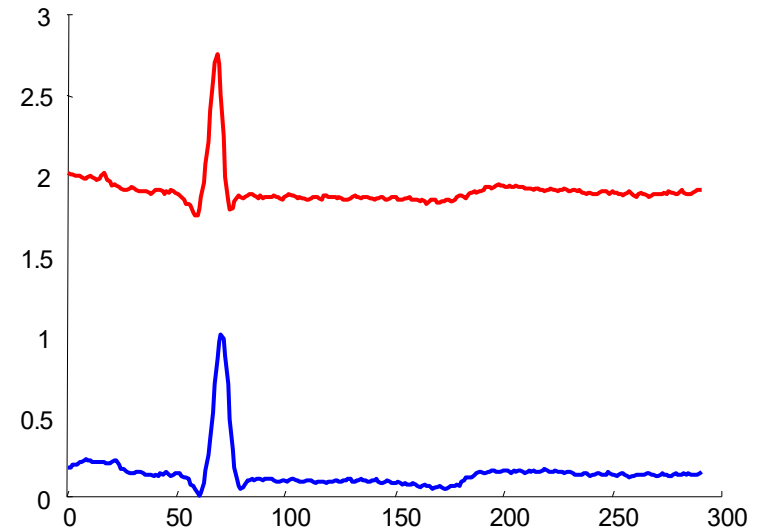
- $T1 = \langle 56, 176, 110, 95 \rangle$
- $T2 = \langle 36, 126, 180, 80 \rangle$

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



# Problems with Euclidean Distance

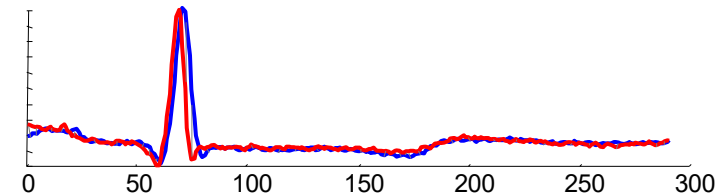
- Euclidean distance is very sensitive to “distortions” in the data.
- These distortions are dangerous and should be removed.
- Most common distortions:
  - Offset Translation
  - Amplitude Scaling
  - Linear Trend
  - Noise
- They can be removed by using the appropriate normalization.



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

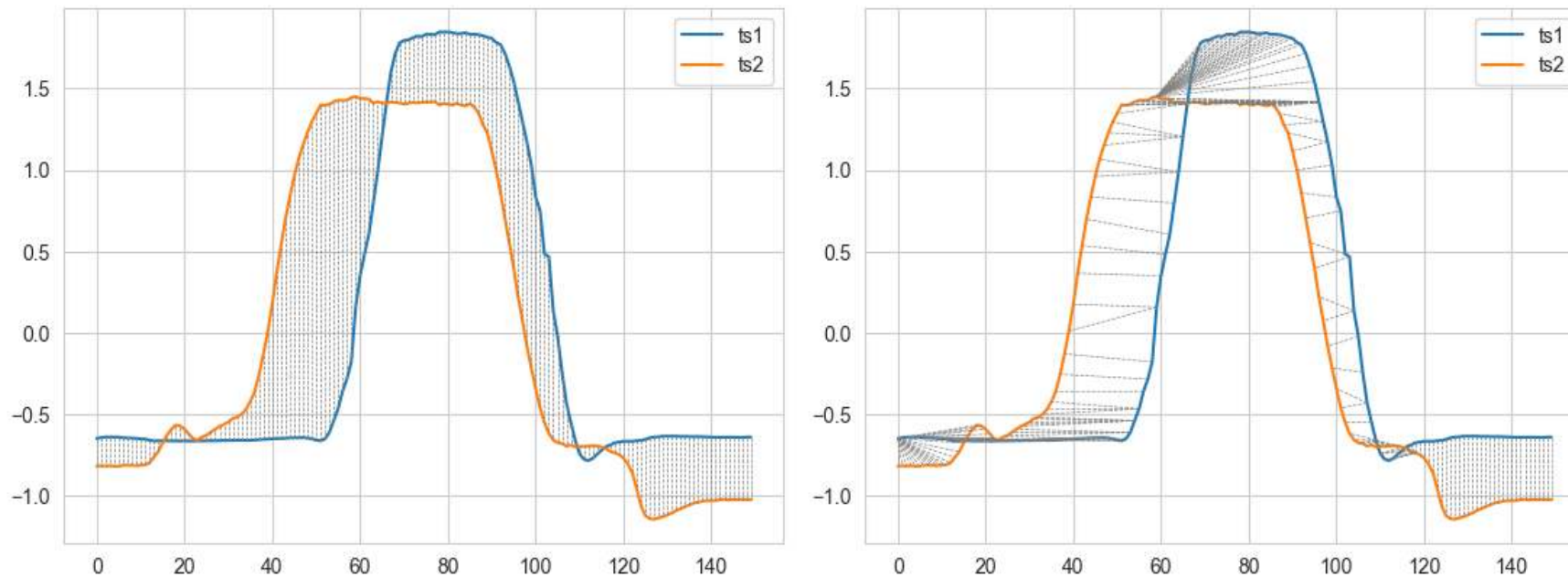
$$C = C - \text{mean}(C)$$

$$D(Q, C)$$



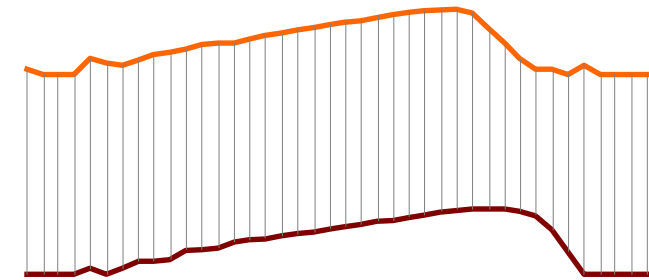
# Further Problems with Euclidean Distance

- Even after normalization, the Euclidean distance may still be unsuitable for some time series domains since it does not allow for distortions along the time axis.

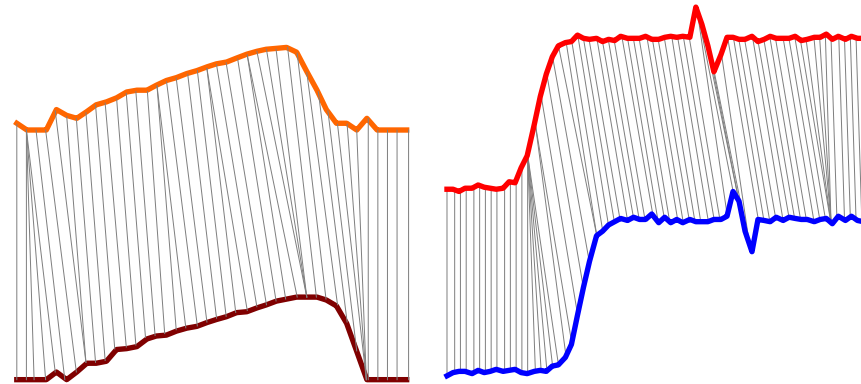
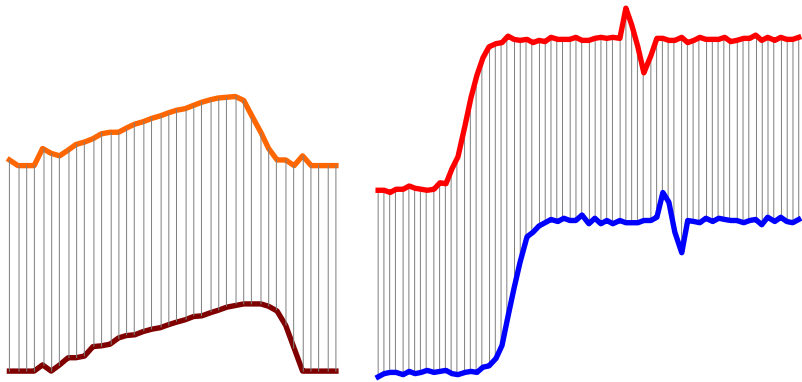


# Dynamic Time Warping

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

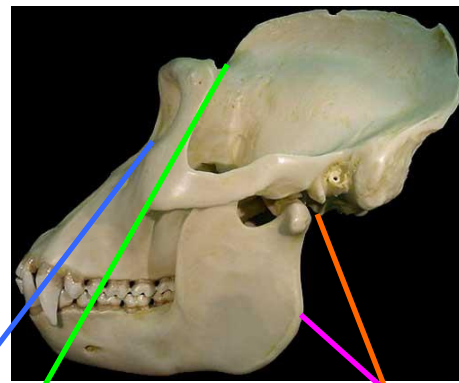


E.g. correspondence of peaks in two similar time series

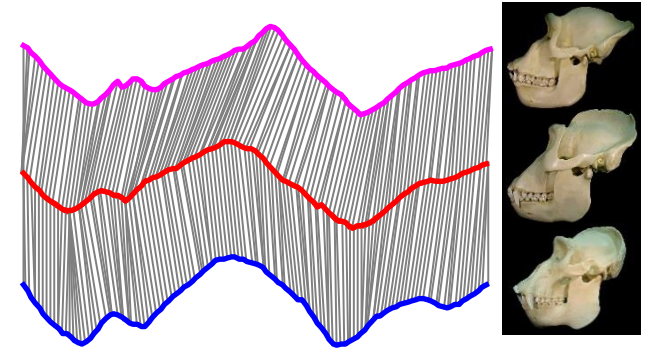
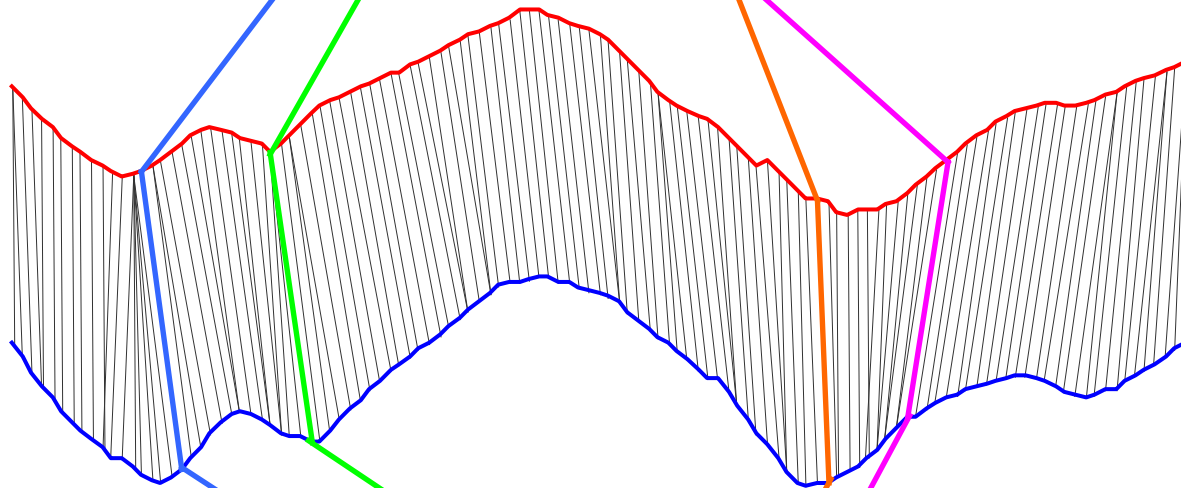
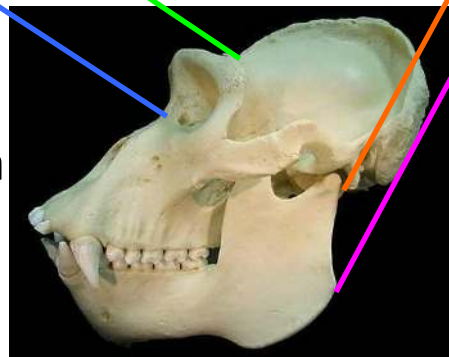


- **Euclidean distance - Fixed Time Axis:** Sequences are aligned “one to one”. Greatly suffers from the misalignment in data.
- **Dynamic Time Warping - Warped Time Axis:** Nonlinear alignments are possible. Can correct misalignments in data.

Lowland Gorilla

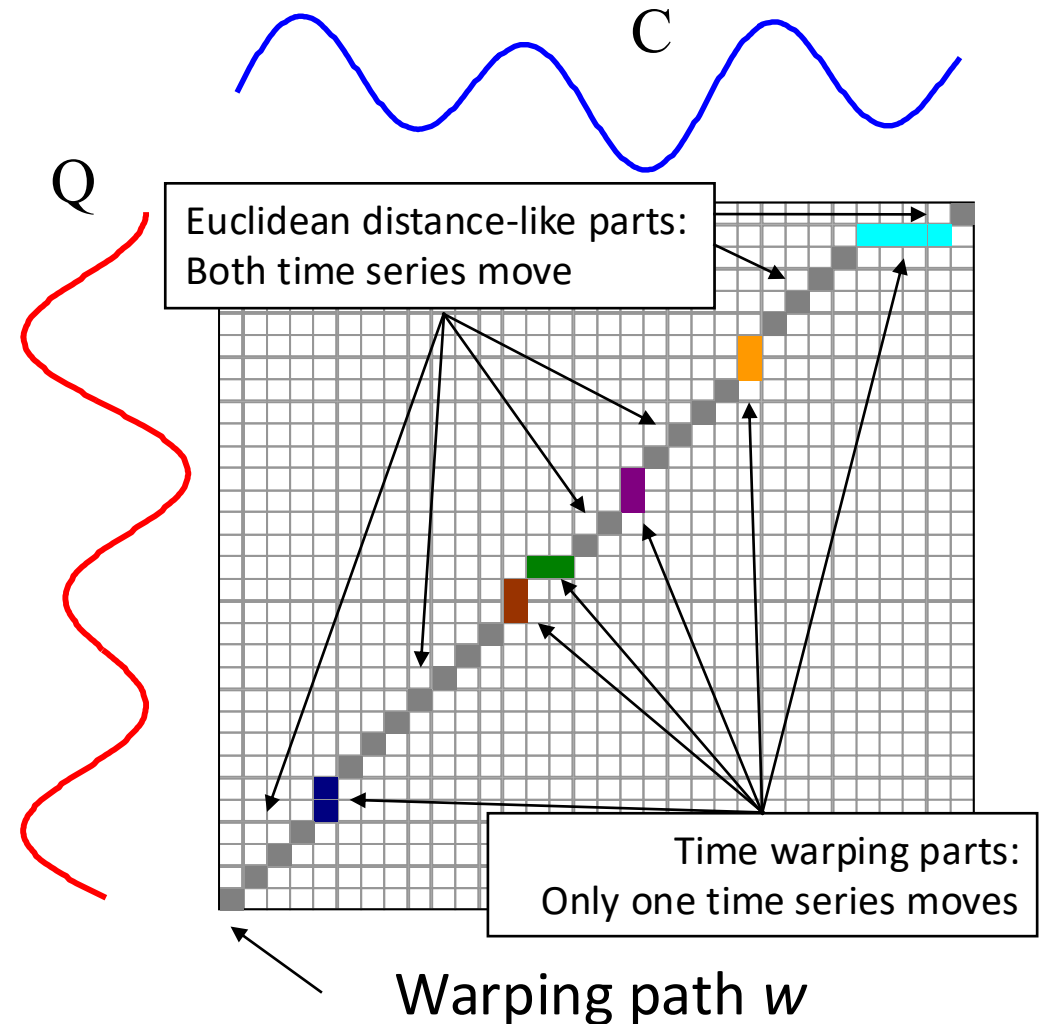
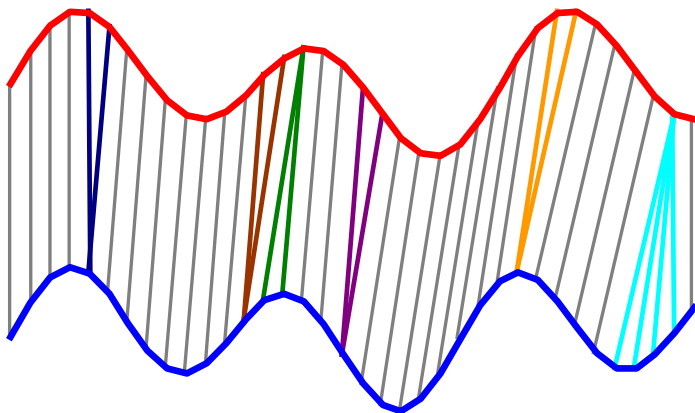


Mountain Gorilla



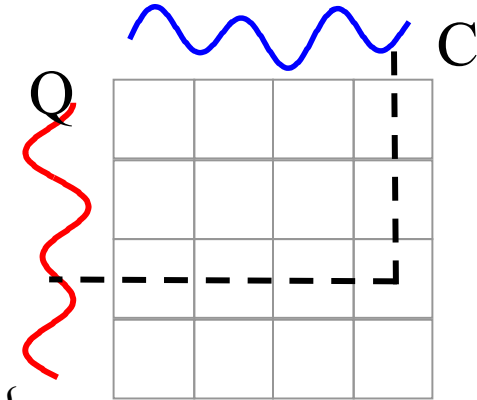
# How is DTW Calculated?

- 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  $(n,m)$



# Dynamic Programming Approach

**Step 1:** Compute the matrix of all *point-to-point* Manhattan distances  $d(q_i, c_j) = |Q_i - C_j|$

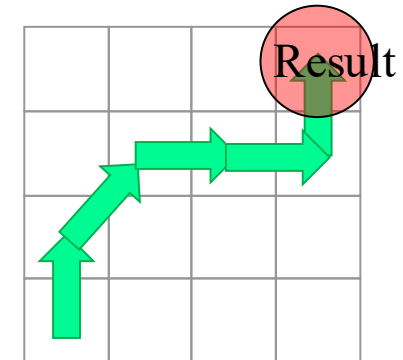


**Step 2:** Compute the cumulative cost matrix as  $\gamma(i,j) = d(q_i, c_j) + \min \{ \gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1) \}$

- Start from cell (1,1)
- Compute (2,1), (3,1), ..., (n,1)
- Repeat for columns 2, 3, ..., n
- Final distance value is in the last cell computed



**Step 3:** find the path with the lowest values, i.e., the best alignment between Q and C



# 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  $\gamma(i,j)$

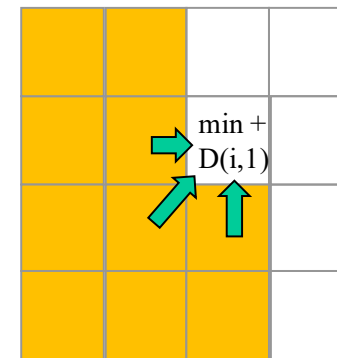
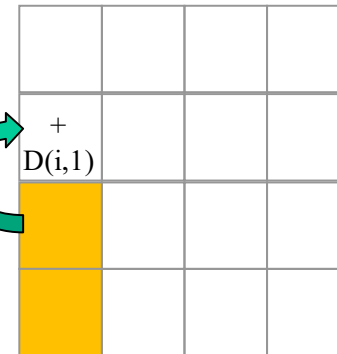
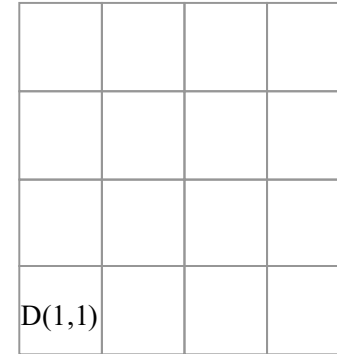
- Start from cell  $(1,1)$

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

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

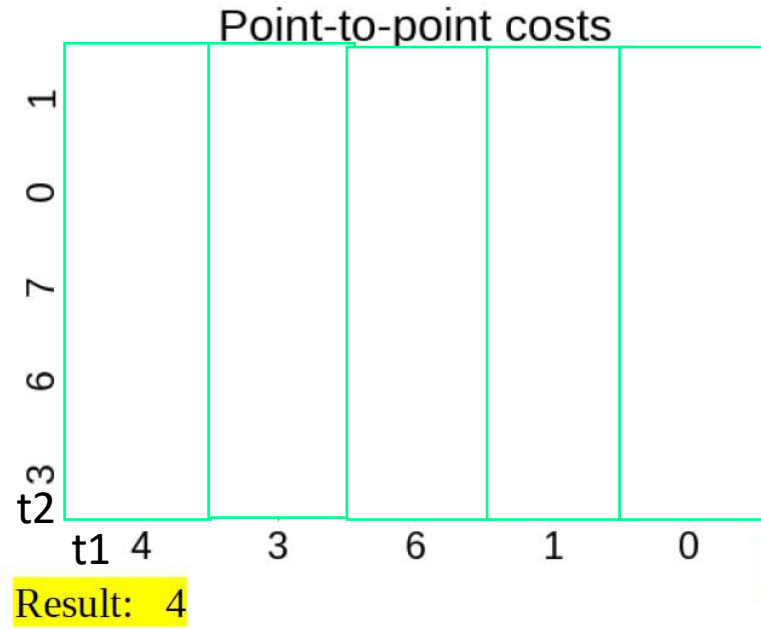
$$\begin{aligned}\gamma(i,1) &= d(q_i, c_1) + \min\{ \gamma(i-1,0), \gamma(i-1,1), \gamma(i,0) \} \\ &= d(q_i, c_1) + \gamma(i-1,1) \\ &= D(i,1) + \gamma(i-1,1)\end{aligned}$$

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



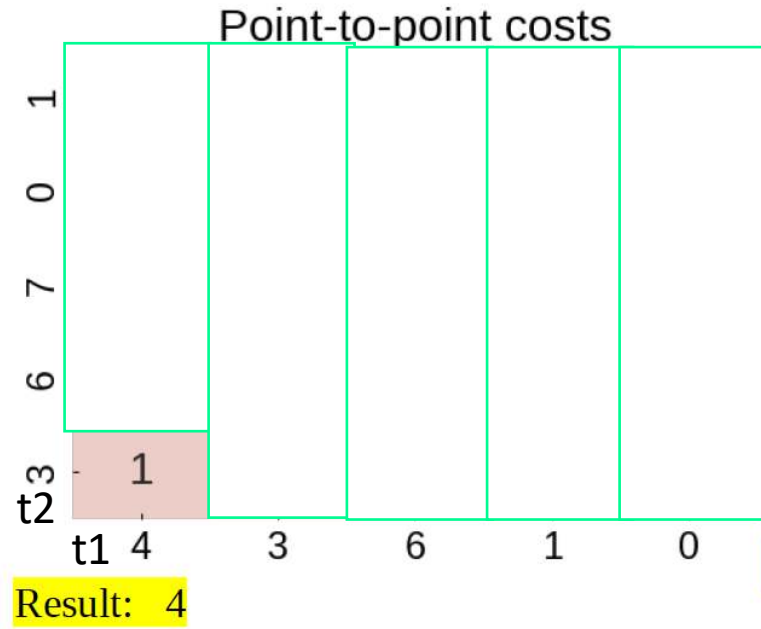
# DTW – Example

|           |                                 |
|-----------|---------------------------------|
| <b>t1</b> | $\langle 4, 3, 6, 1, 0 \rangle$ |
| <b>t2</b> | $\langle 3, 6, 7, 0, 1 \rangle$ |



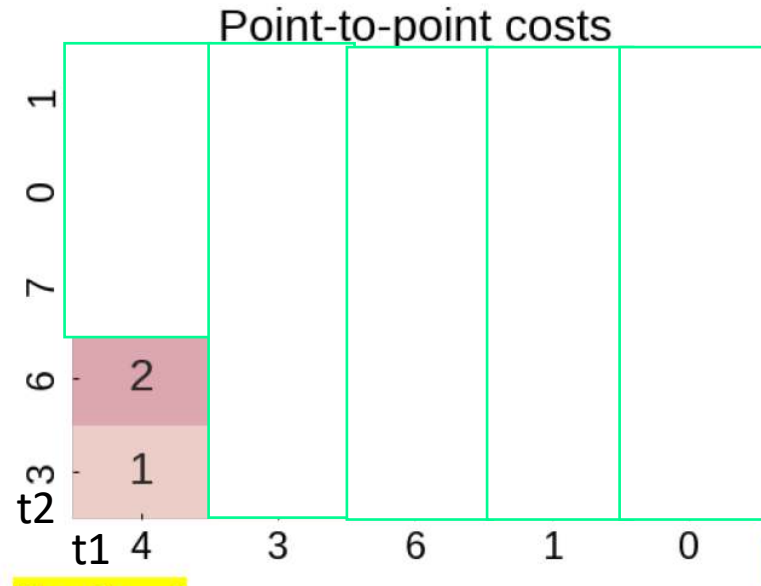
# DTW – Example

|           |                                 |
|-----------|---------------------------------|
| <b>t1</b> | $\langle 4, 3, 6, 1, 0 \rangle$ |
| <b>t2</b> | $\langle 3, 6, 7, 0, 1 \rangle$ |



# DTW – Example

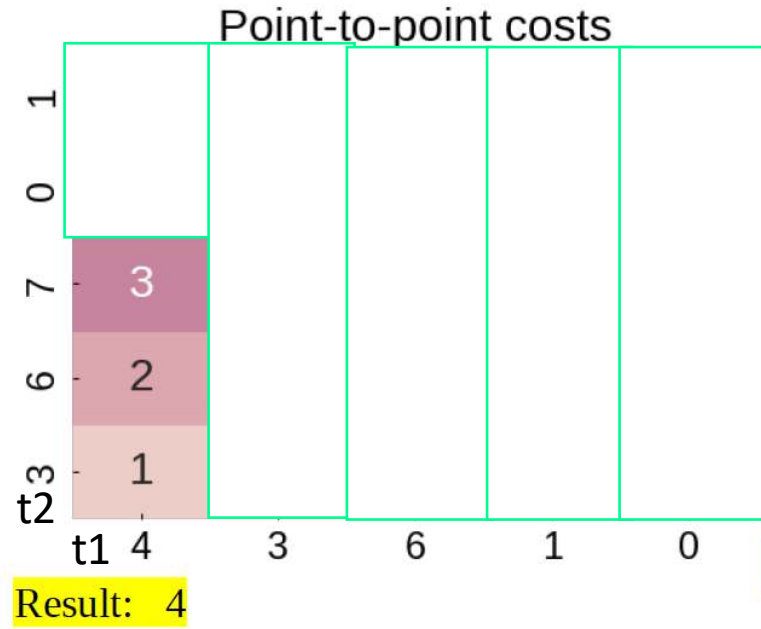
|           |                                 |
|-----------|---------------------------------|
| <b>t1</b> | $\langle 4, 3, 6, 1, 0 \rangle$ |
| <b>t2</b> | $\langle 3, 6, 7, 0, 1 \rangle$ |



Result: 4

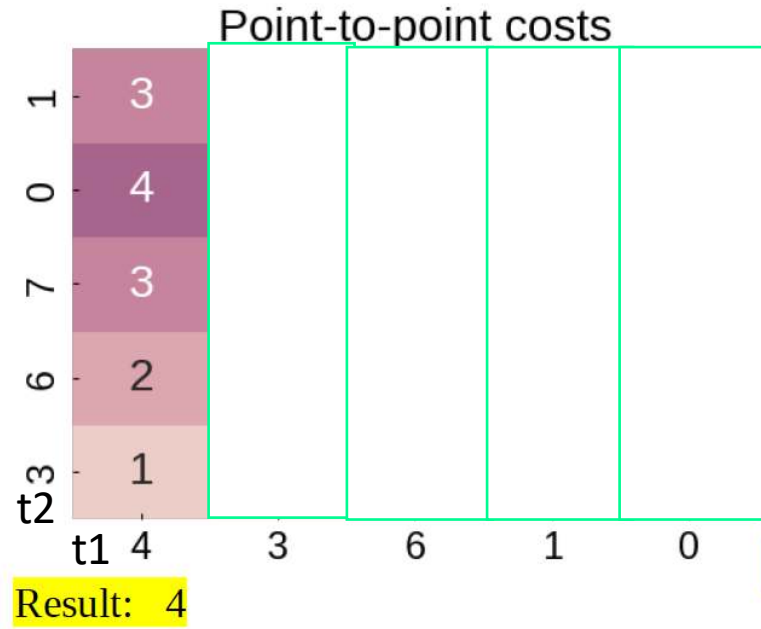
# DTW – Example

|           |                                 |
|-----------|---------------------------------|
| <b>t1</b> | $\langle 4, 3, 6, 1, 0 \rangle$ |
| <b>t2</b> | $\langle 3, 6, 7, 0, 1 \rangle$ |



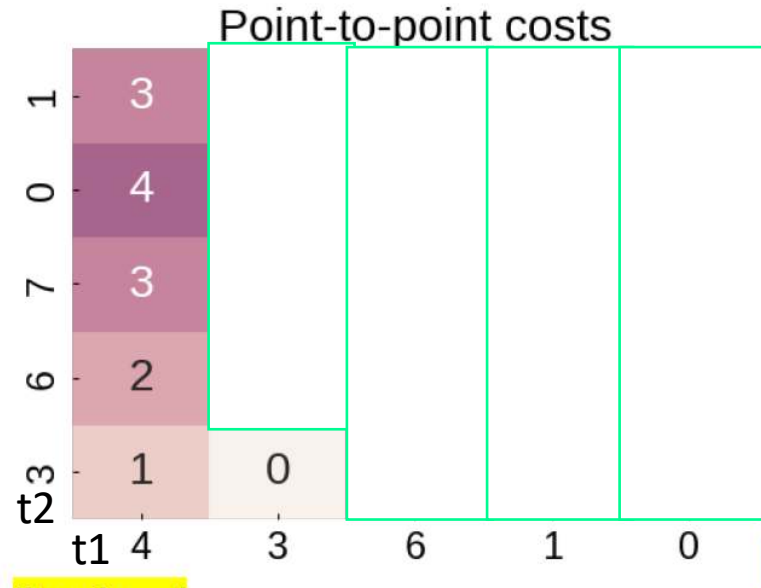
# DTW – Example

|           |                                 |
|-----------|---------------------------------|
| <b>t1</b> | $\langle 4, 3, 6, 1, 0 \rangle$ |
| <b>t2</b> | $\langle 3, 6, 7, 0, 1 \rangle$ |



# DTW – Example

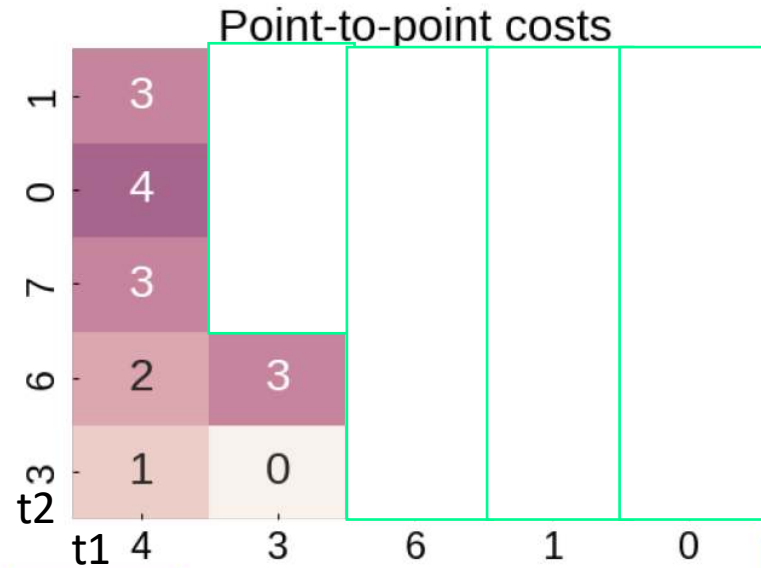
|           |                                 |
|-----------|---------------------------------|
| <b>t1</b> | $\langle 4, 3, 6, 1, 0 \rangle$ |
| <b>t2</b> | $\langle 3, 6, 7, 0, 1 \rangle$ |



Result: 4

# DTW – Example

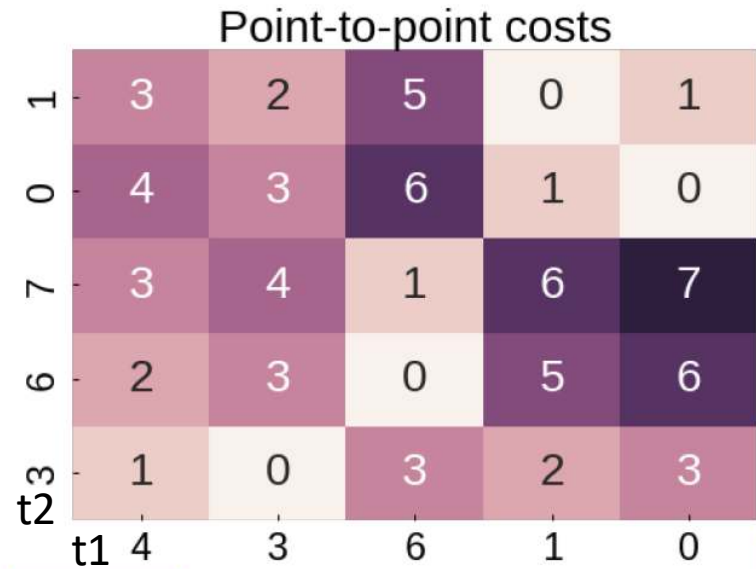
|           |                                 |
|-----------|---------------------------------|
| <b>t1</b> | $\langle 4, 3, 6, 1, 0 \rangle$ |
| <b>t2</b> | $\langle 3, 6, 7, 0, 1 \rangle$ |



Result: 4

# DTW – Example

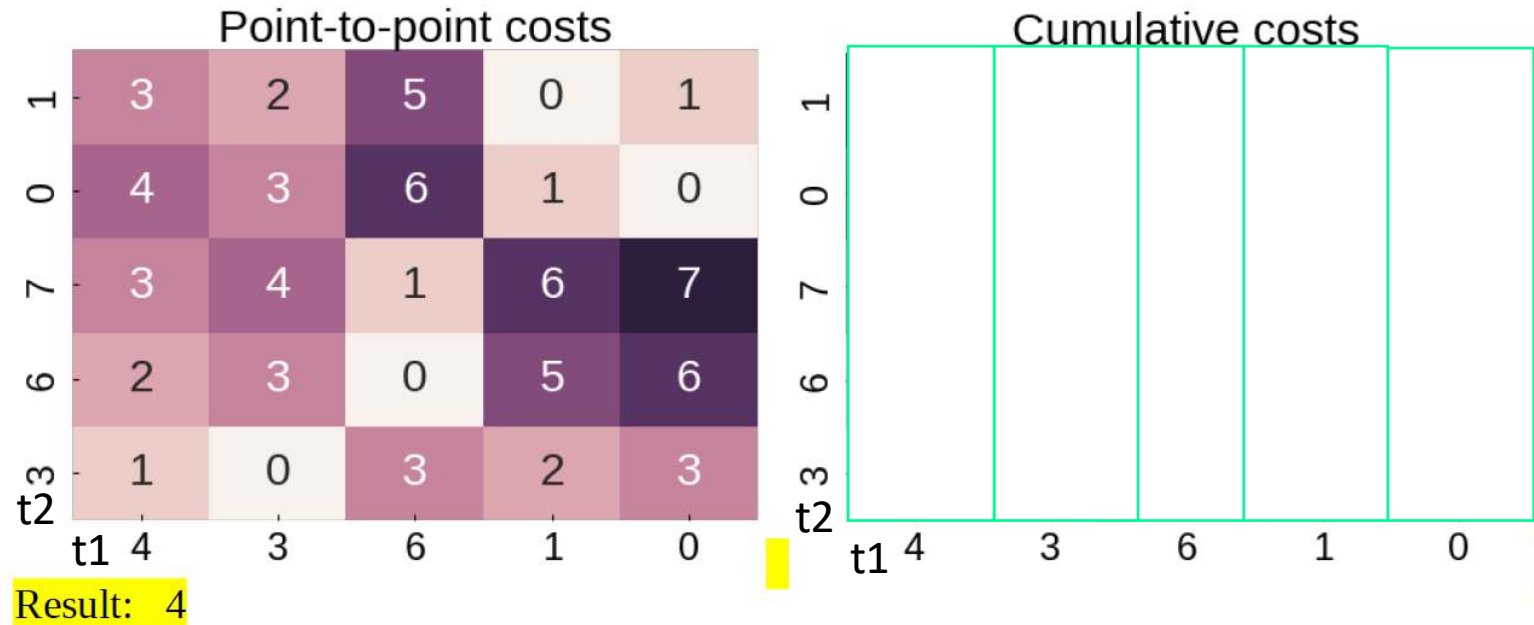
|    |                   |
|----|-------------------|
| t1 | < 4, 3, 6, 1, 0 > |
| t2 | < 3, 6, 7, 0, 1 > |



Result: 4

# DTW – Example

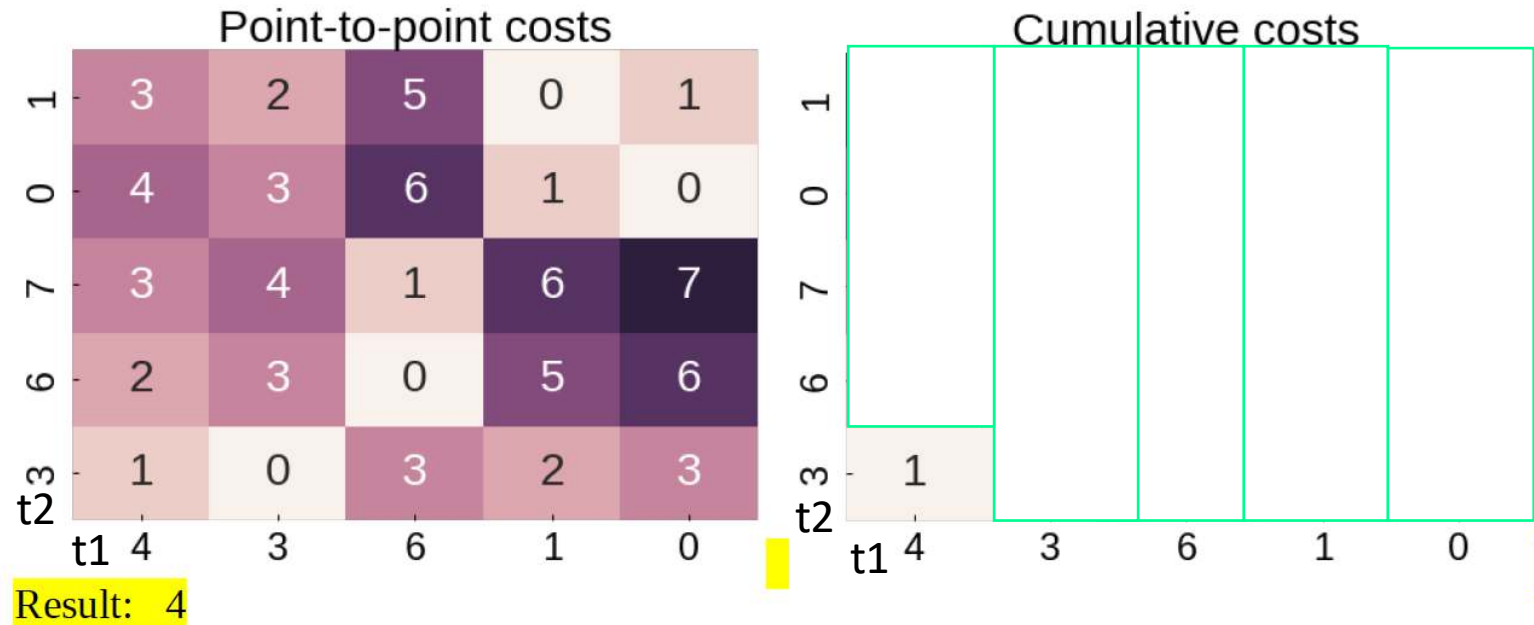
|           |                   |
|-----------|-------------------|
| <b>t1</b> | < 4, 3, 6, 1, 0 > |
| <b>t2</b> | < 3, 6, 7, 0, 1 > |



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

# DTW – Example

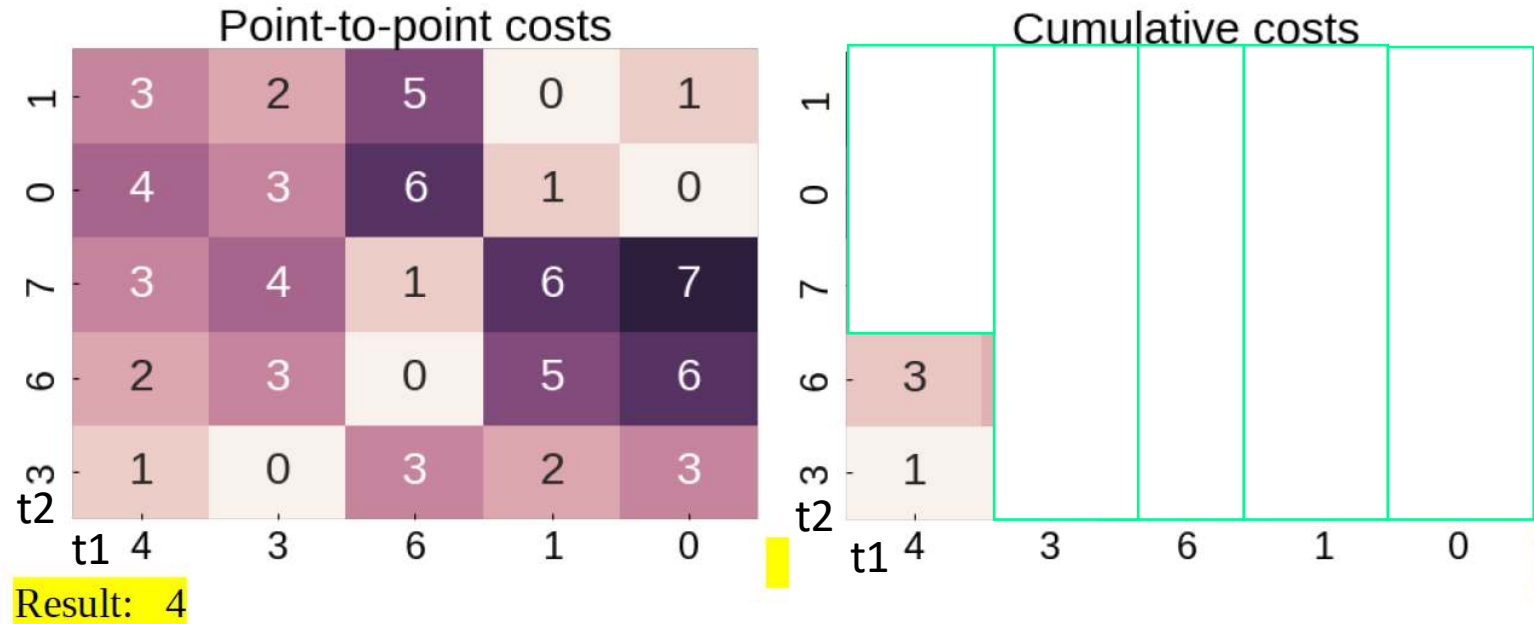
|           |                   |
|-----------|-------------------|
| <b>t1</b> | < 4, 3, 6, 1, 0 > |
| <b>t2</b> | < 3, 6, 7, 0, 1 > |



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

# DTW – Example

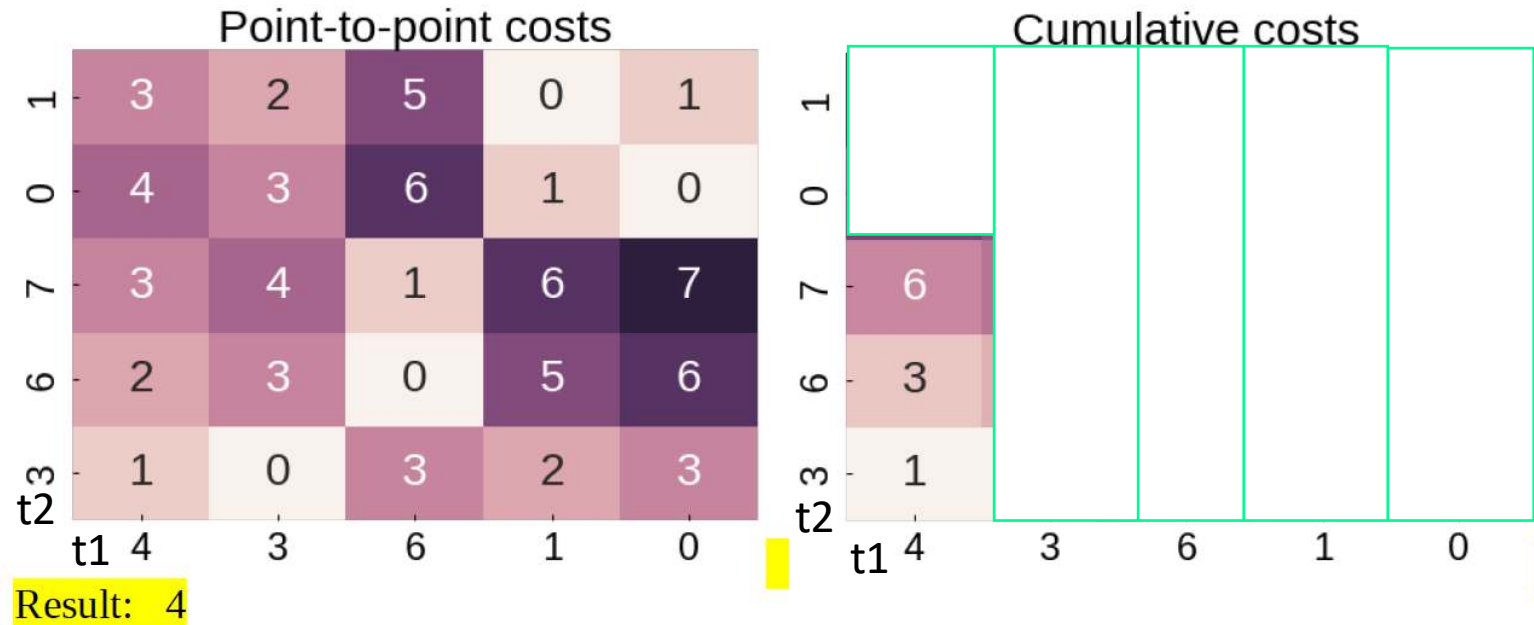
|           |                   |
|-----------|-------------------|
| <b>t1</b> | < 4, 3, 6, 1, 0 > |
| <b>t2</b> | < 3, 6, 7, 0, 1 > |



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

# DTW – Example

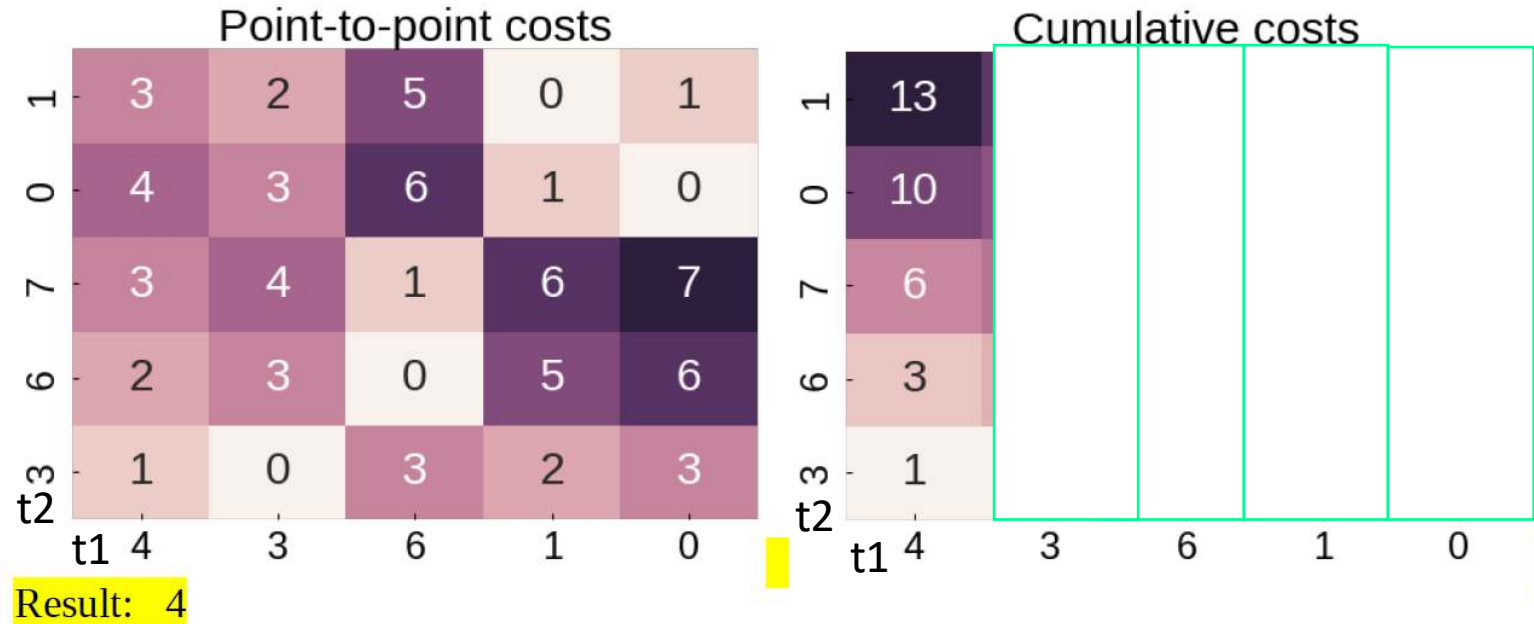
|           |                   |
|-----------|-------------------|
| <b>t1</b> | < 4, 3, 6, 1, 0 > |
| <b>t2</b> | < 3, 6, 7, 0, 1 > |



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

# DTW – Example

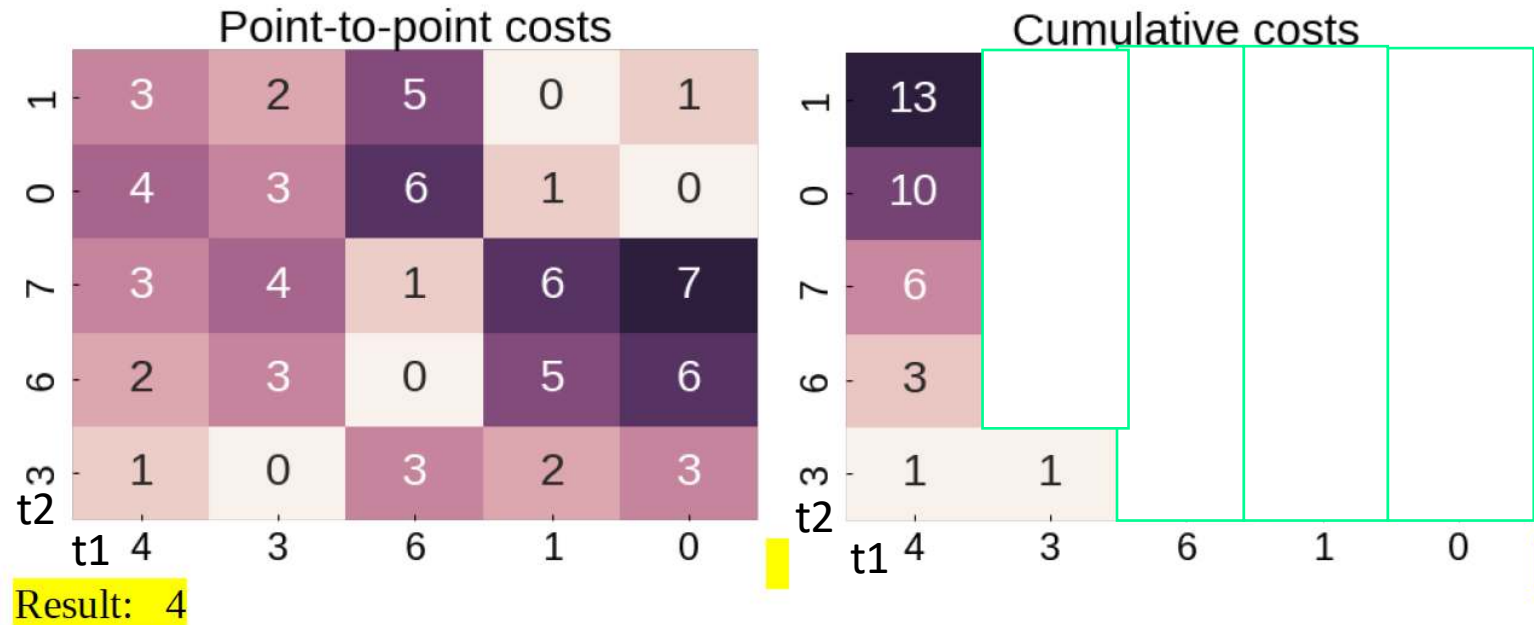
|           |                   |
|-----------|-------------------|
| <b>t1</b> | < 4, 3, 6, 1, 0 > |
| <b>t2</b> | < 3, 6, 7, 0, 1 > |



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

# DTW – Example

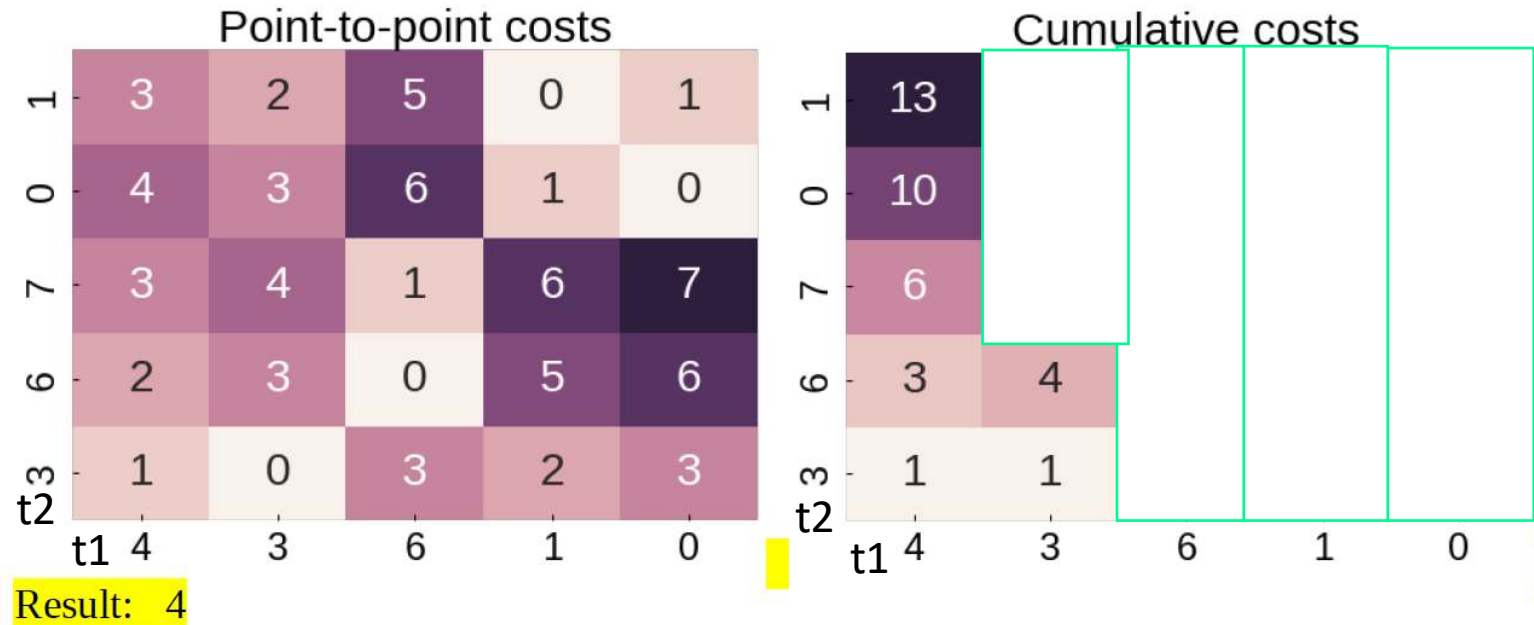
|           |                   |
|-----------|-------------------|
| <b>t1</b> | < 4, 3, 6, 1, 0 > |
| <b>t2</b> | < 3, 6, 7, 0, 1 > |



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

# DTW – Example

|           |                   |
|-----------|-------------------|
| <b>t1</b> | < 4, 3, 6, 1, 0 > |
| <b>t2</b> | < 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 – Example

|           |                   |
|-----------|-------------------|
| <b>t1</b> | < 4, 3, 6, 1, 0 > |
| <b>t2</b> | < 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 – Example

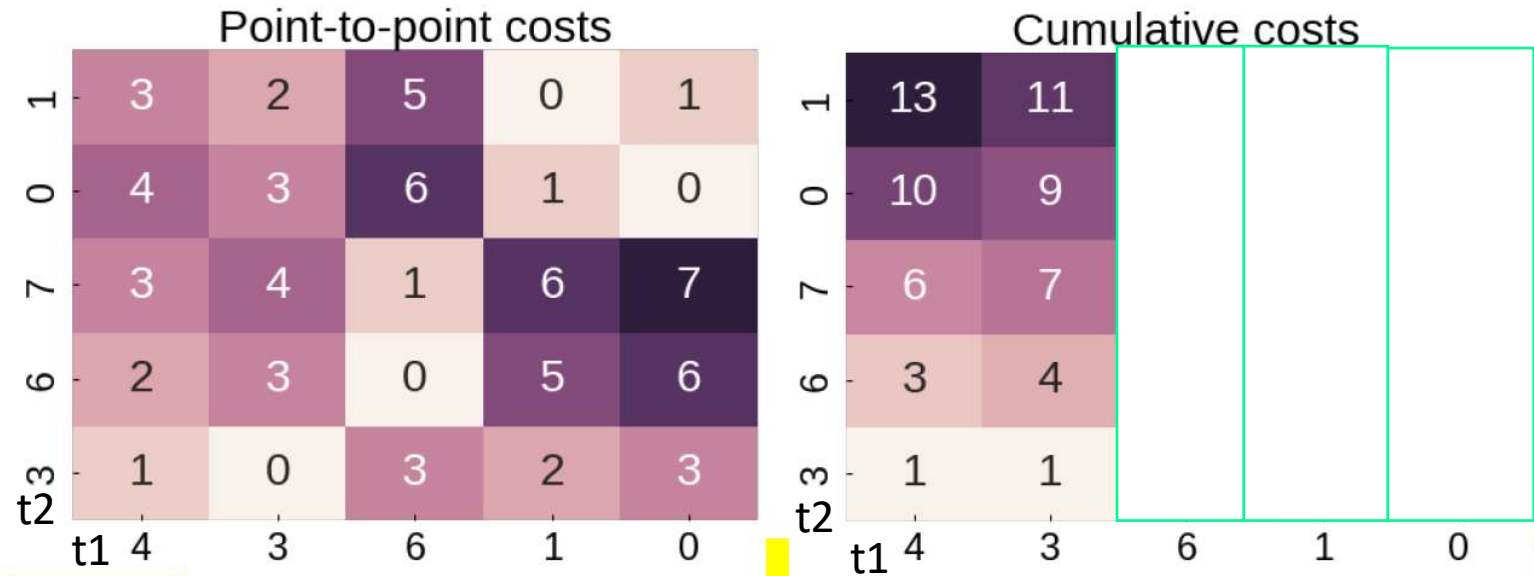
|           |                   |
|-----------|-------------------|
| <b>t1</b> | < 4, 3, 6, 1, 0 > |
| <b>t2</b> | < 3, 6, 7, 0, 1 > |



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

# DTW – Example

|           |                   |
|-----------|-------------------|
| <b>t1</b> | < 4, 3, 6, 1, 0 > |
| <b>t2</b> | < 3, 6, 7, 0, 1 > |



Result: 4

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

# DTW – Example

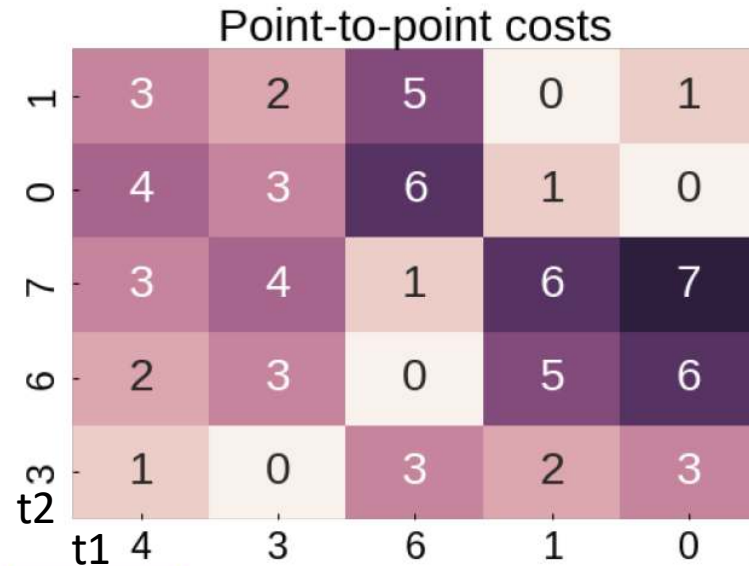
|           |                   |
|-----------|-------------------|
| <b>t1</b> | < 4, 3, 6, 1, 0 > |
| <b>t2</b> | < 3, 6, 7, 0, 1 > |



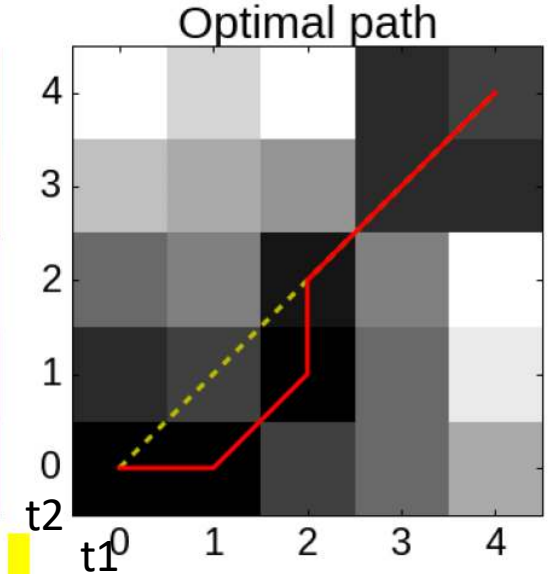
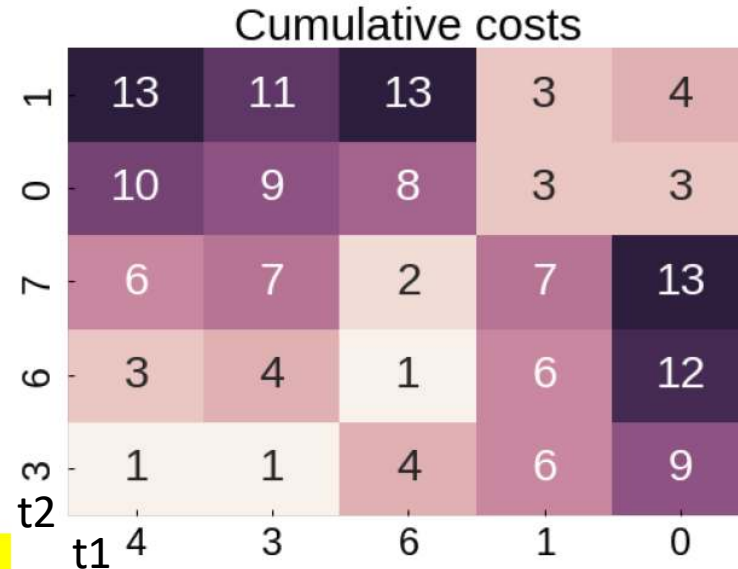
$$\gamma(i,j) = d(q_i, c_j) + \min \{ \gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1) \}$$

# DTW – Example

|           |                                 |
|-----------|---------------------------------|
| <b>t1</b> | $\langle 4, 3, 6, 1, 0 \rangle$ |
| <b>t2</b> | $\langle 3, 6, 7, 0, 1 \rangle$ |

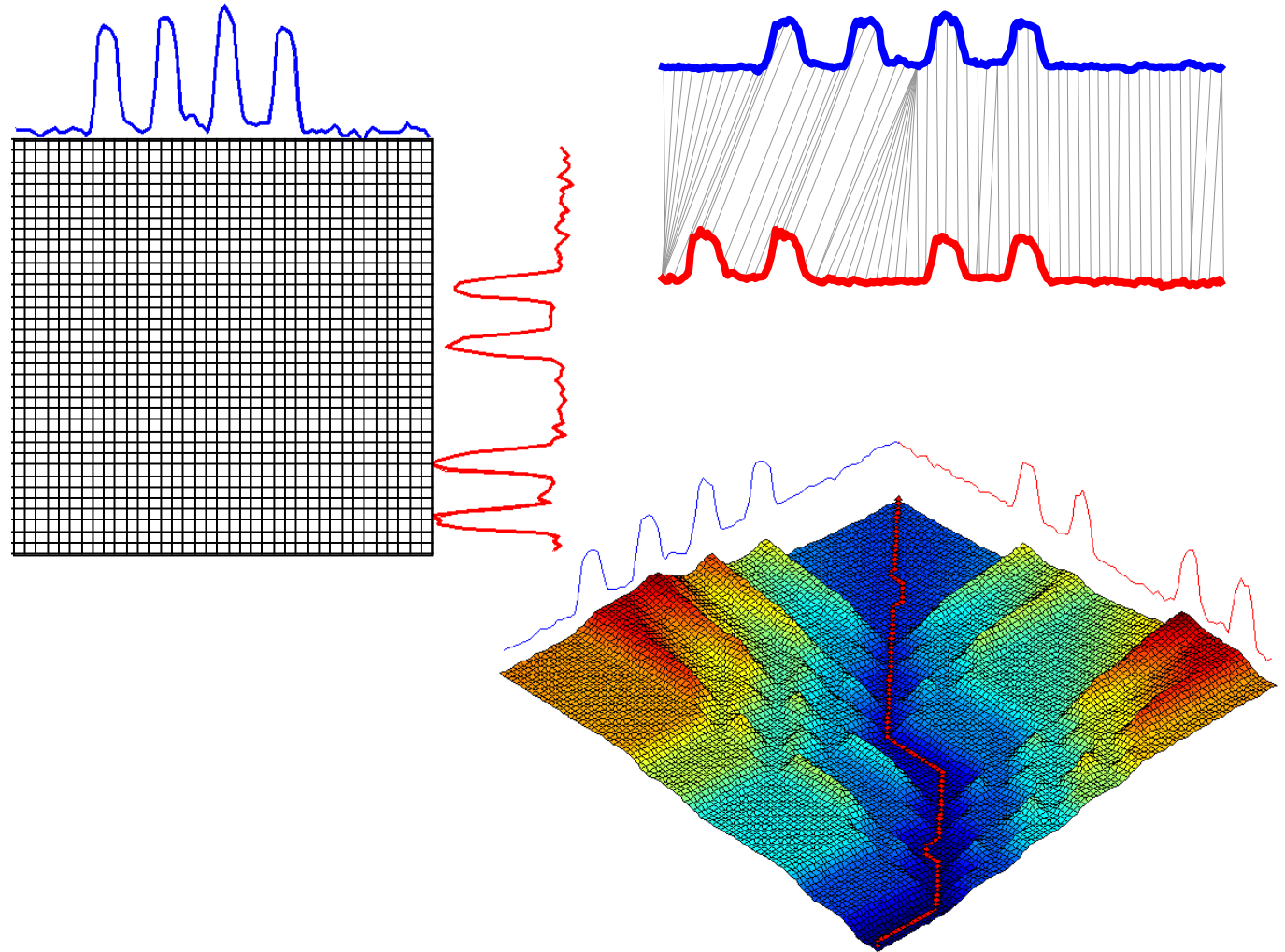


Result: 4



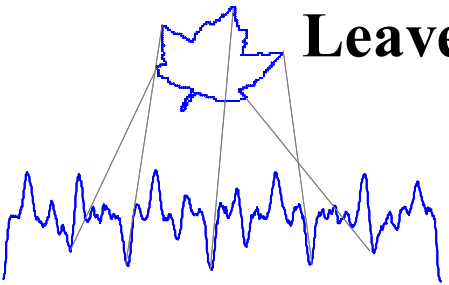
# DTW – A Real Example

- 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

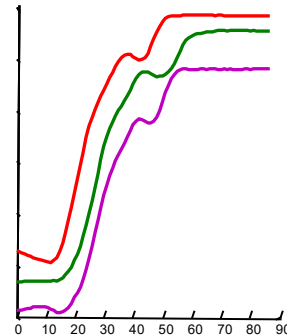
**Leaves**



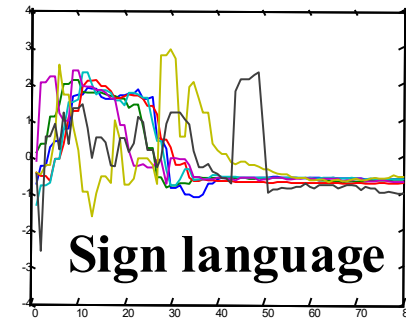
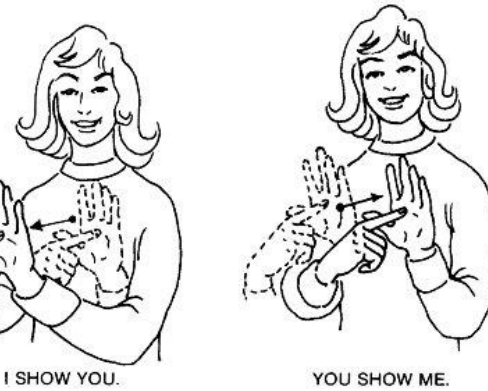
**Faces**



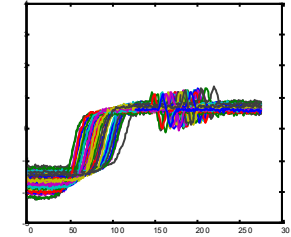
**Gun**



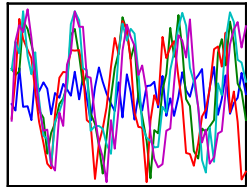
**Sign language**



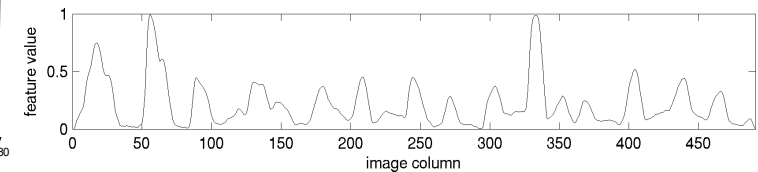
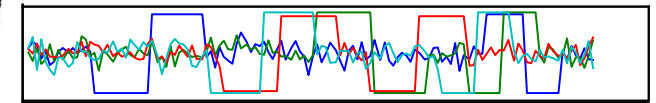
**Trace**



**Control**



**2-Patterns**



**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

| Accuracy            |           |      |
|---------------------|-----------|------|
| Dataset             | Euclidean | DTW  |
| Word Spotting       | 0.95      | 0.99 |
| Sign language       | 0.71      | 0.74 |
| GUN                 | 0.95      | 0.99 |
| Nuclear Trace       | 0.89      | 1.00 |
| Leaves <sup>#</sup> | 0.67      | 0.96 |
| (4) Faces           | 0.94      | 0.97 |
| Control Chart*      | 0.93      | 1.00 |
| 2-Patterns          | 0.99      | 1.00 |

# 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 |

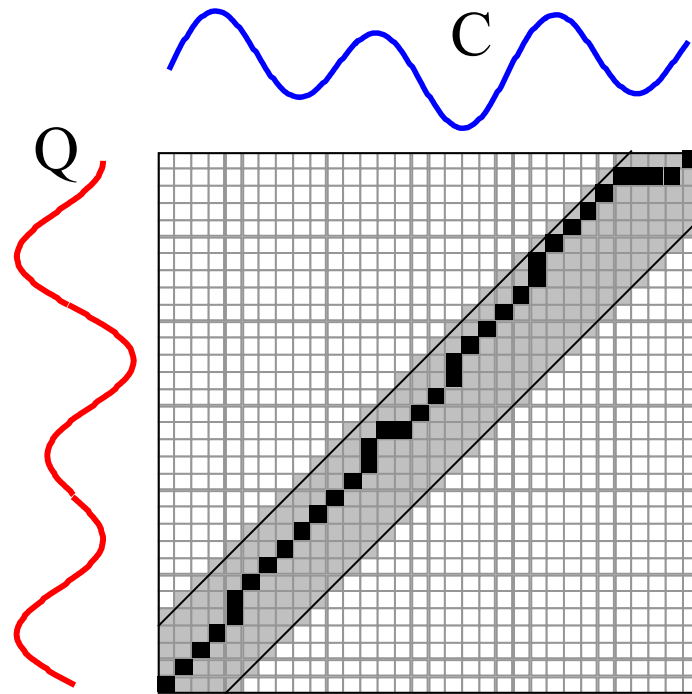
# Problems with Dynamic Time Warping

---

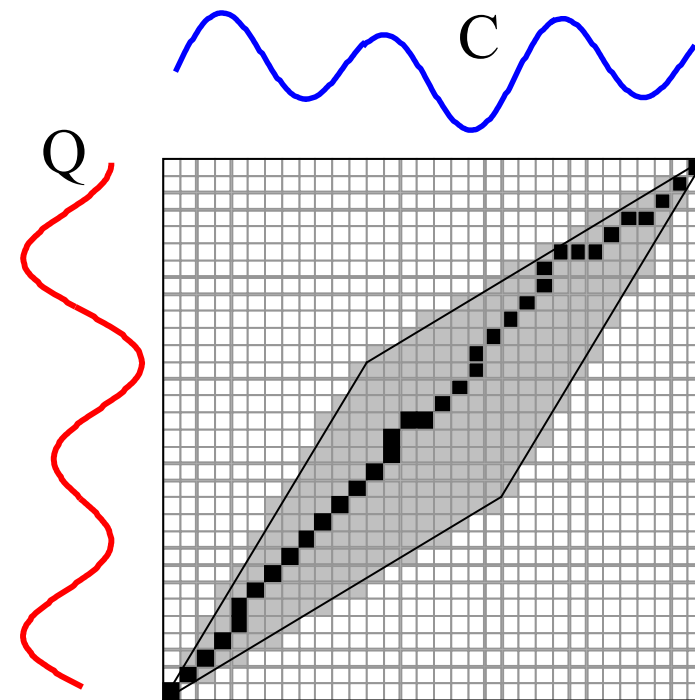
- Dynamic Time Warping gives much better results than Euclidean distance on many problems.
- Dynamic Time Warping is very very slow to calculate!
- Is there anything we can do to speed up similarity search under DTW?

# Global Constraints

- Slightly speed up the calculations
- Prevent pathological warpings



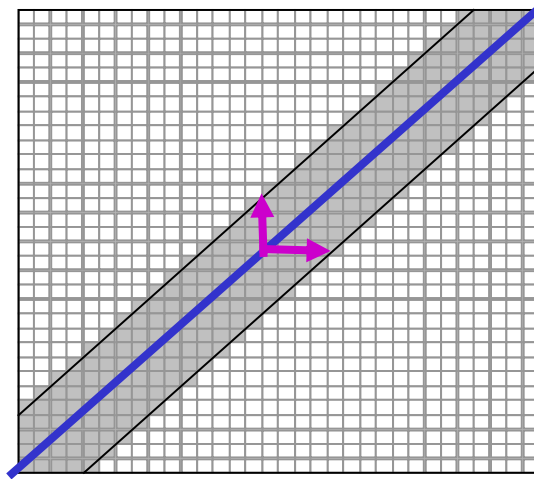
Sakoe-Chiba Band



Itakura Parallelogram

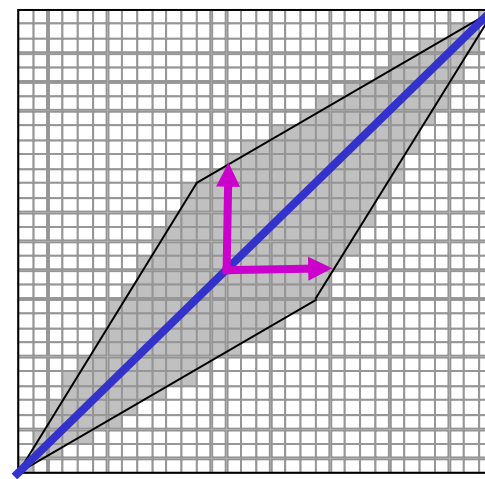
# Global Constraints

- 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.



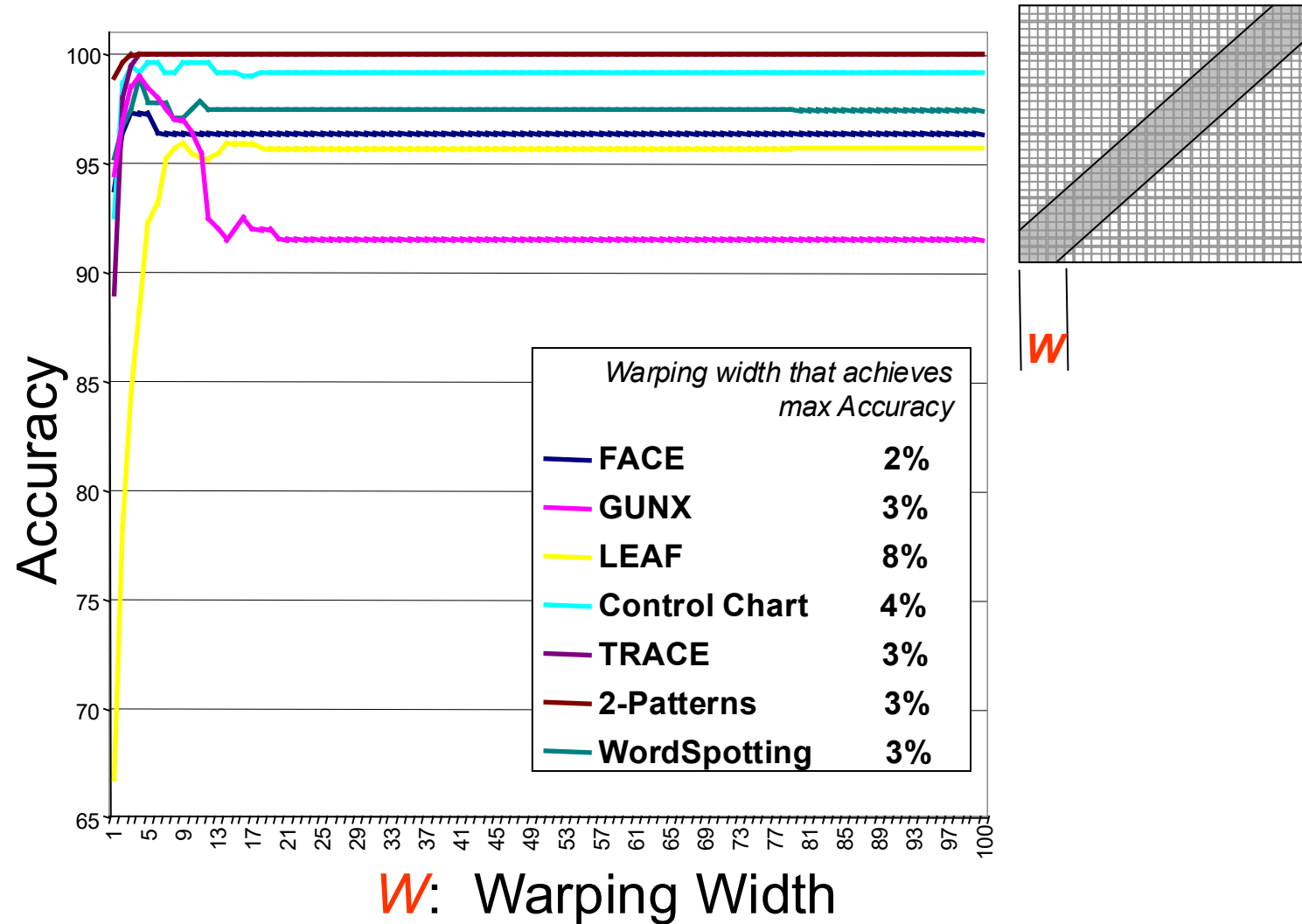
Sakoe-Chiba Band

$r_i$



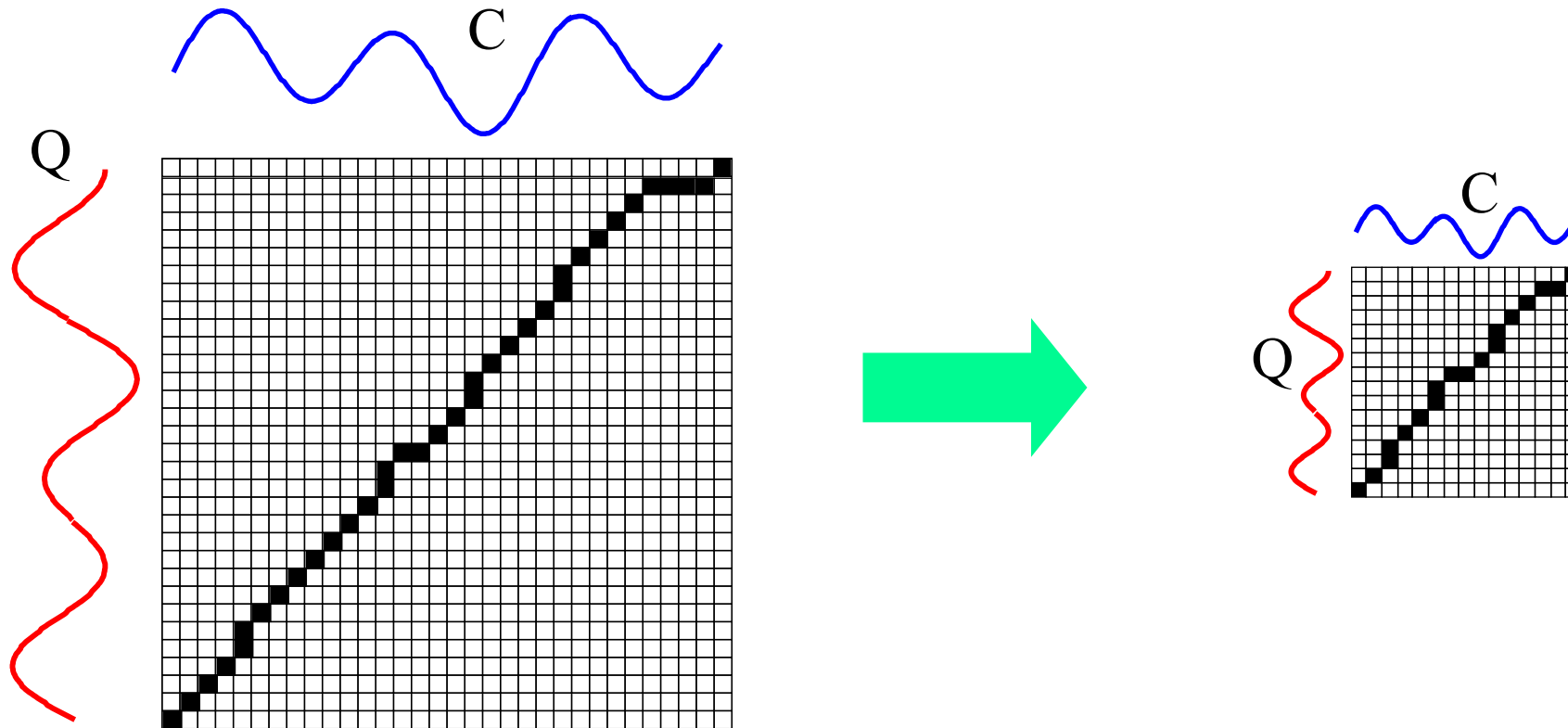
Itakura Parallelogram

# Accuracy vs. Width of Warping Window



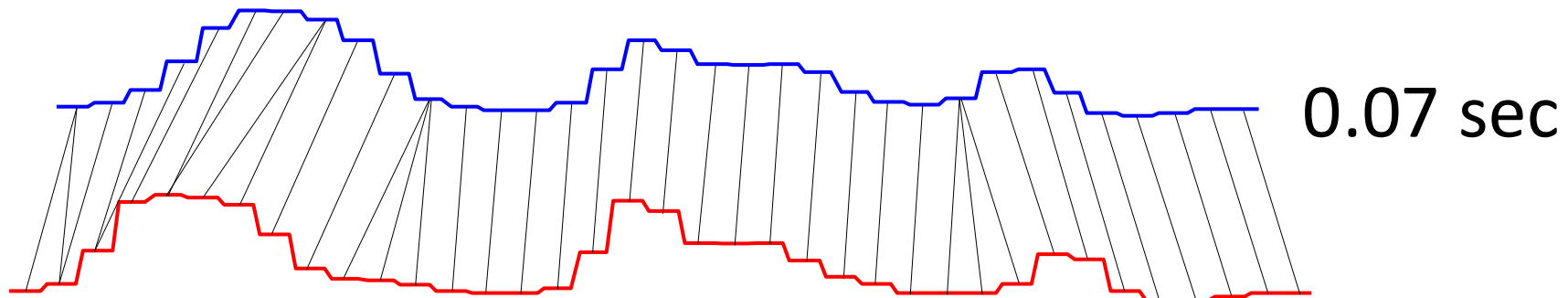
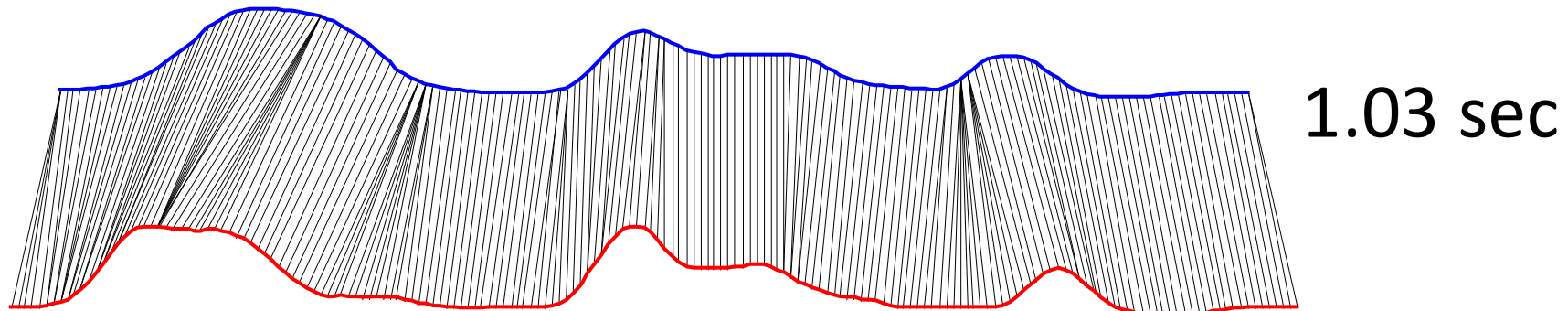
# Fast Approximations to DTW

- 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.



# Distances and Normalizations

---

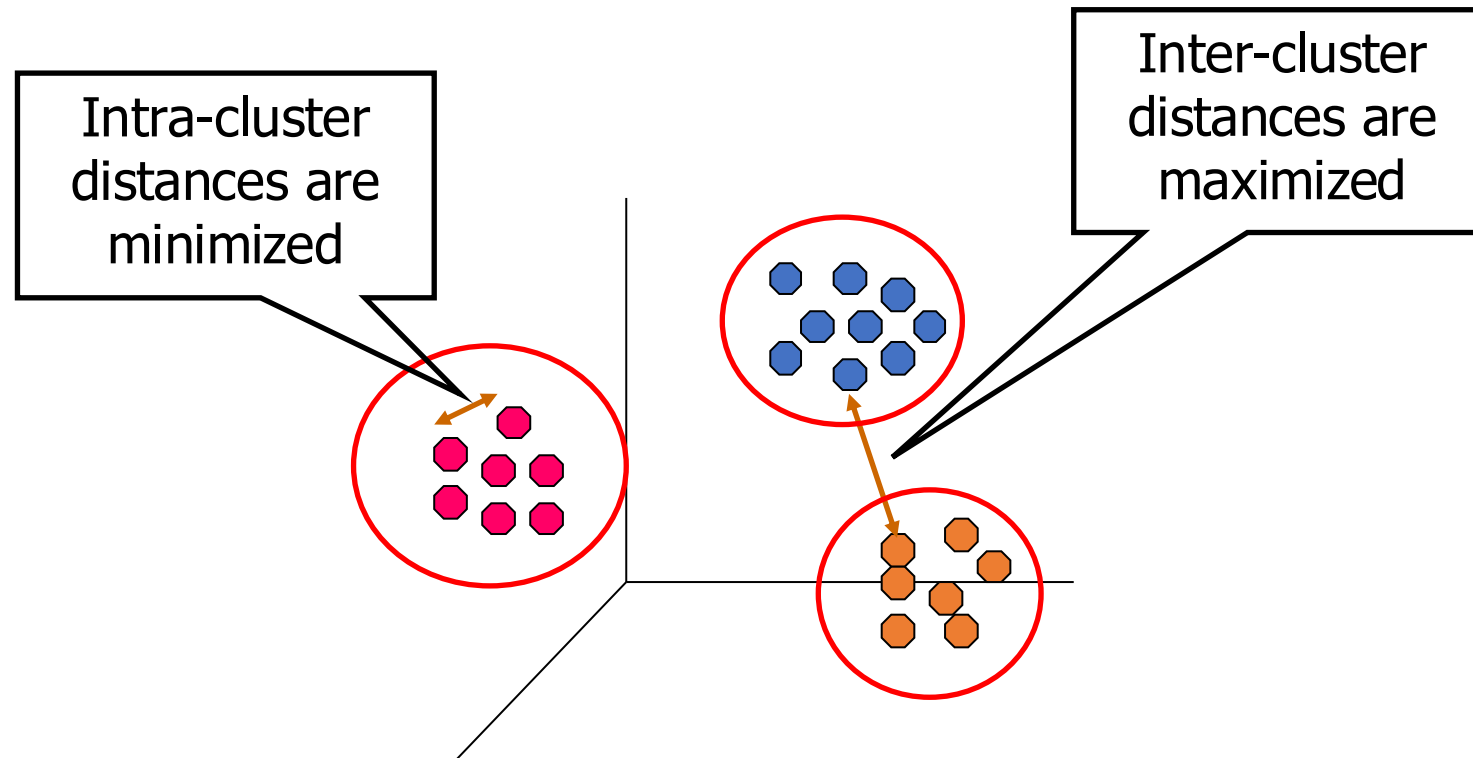
- If a distance measure is intended to capture shape-based similarity, it is important to decide whether the level of the time series should be part of the comparison.
- For downstream tasks, the choice should depend on which information is relevant for the task.
- This kind of reasoning applies also to other features of the TS.

# Time Series Clustering

---

# What is Cluster Analysis?

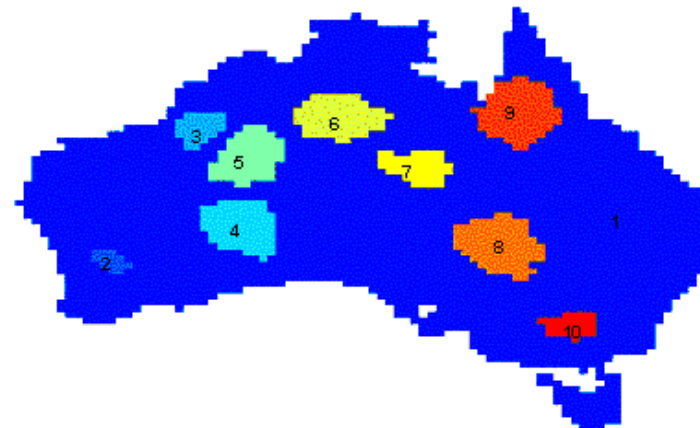
- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups.



# Applications of Cluster Analysis

- **Understanding**
  - Group related documents for browsing, group genes and proteins that have similar functionality, or group stocks with similar price fluctuations
- **Summarization**
  - Reduce the size of large data sets

|          | <i>Discovered Clusters</i>                                                                                                                                                                                                                      | <i>Industry Group</i> |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| <b>1</b> | Applied-Matl-DOWN, Bay-Network-DOWN, 3-COM-DOWN, Cabletron-Sys-DOWN, CISCO-DOWN, HP-DOWN, DSC-Comm-DOWN, INTEL-DOWN, LSI-Logic-DOWN, Micron-Tech-DOWN, Texas-Inst-DOWN, Tellabs-Inc-DOWN, Natl-Semiconduct-DOWN, Oracl-DOWN, SGI-DOWN, Sun-DOWN | Technology1-DOWN      |
| <b>2</b> | Apple-Comp-DOWN, Autodesk-DOWN, DEC-DOWN, ADV-Micro-Device-DOWN, Andrew-Corp-DOWN, Computer-Assoc-DOWN, Circuit-City-DOWN, Compaq-DOWN, EMC-Corp-DOWN, Gen-Inst-DOWN, Motorola-DOWN, Microsoft-DOWN, Scientific-Atl-DOWN                        | Technology2-DOWN      |
| <b>3</b> | Fannie-Mae-DOWN, Fed-Home-Loan-DOWN, MBNA-Corp-DOWN, Morgan-Stanley-DOWN                                                                                                                                                                        | Financial-DOWN        |
| <b>4</b> | Baker-Hughes-UP, Dresser-Inds-UP, Halliburton-HLD-UP, Louisiana-Land-UP, Phillips-Petro-UP, Unocal-UP, Schlumberger-UP                                                                                                                          | Oil-UP                |



Clustering precipitation  
in Australia

# What is not Cluster Analysis?

---

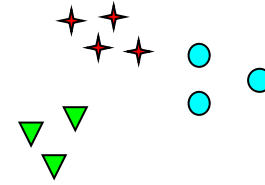
- Simple segmentation
  - Dividing students into different registration groups alphabetically, by last name
- Results of a query
  - Groupings are a result of an external specification
  - Clustering is a grouping of objects based on the data
- Supervised classification
  - Have class label information
- Association Analysis
  - Local vs. global connections

# Notion of a Cluster can be Ambiguous

---



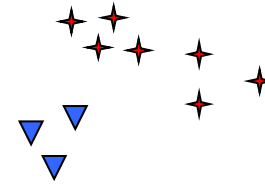
How many clusters?



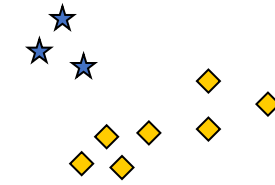
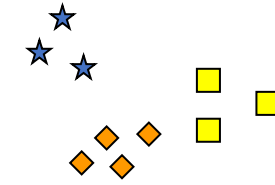
Six Clusters



Two Clusters



Four Clusters



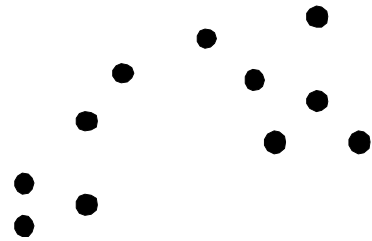
# Types of Clusterings

---

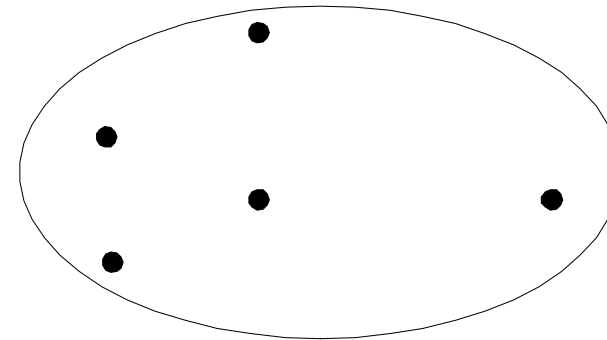
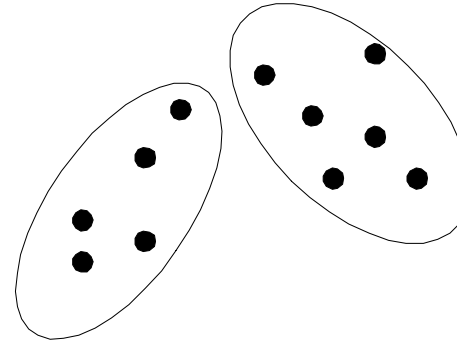
- A **clustering** is a set of clusters
- Important distinction between **hierarchical** and **partitional** sets of clusters
- Partitional Clustering
  - A division of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- Hierarchical clustering
  - A set of nested clusters organized as a hierarchical tree

# Partitional Clustering

---



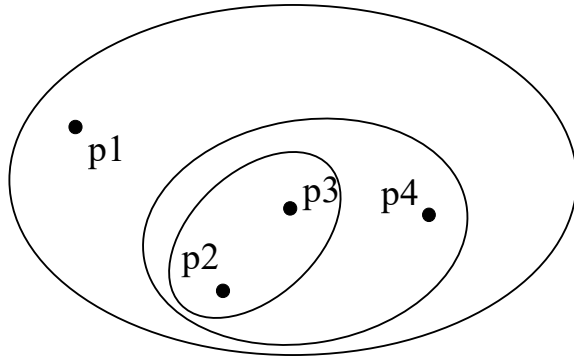
**Original Points**



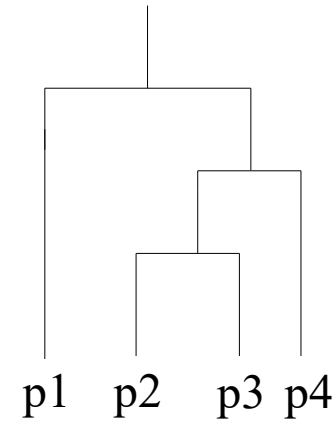
**A Partitional Clustering**

# Hierarchical Clustering

---



**Traditional Hierarchical Clustering**



**Traditional Dendrogram**

# Types of Clusters: Objective Function

---

- Clusters Defined by an Objective Function
  - Finds clusters that **minimize** or **maximize an objective function**.
  - Enumerate all possible ways of dividing the points into clusters and evaluate the 'goodness' of each potential set of clusters by using the given objective function. (NP Hard)
  - Can have global or local objectives.
    - Hierarchical clustering algorithms typically have local objectives
    - Partitional algorithms typically have global objectives

# Characteristics of the Input Data Are Important

---

- Type of proximity or density measure
  - Central to clustering
  - Depends on data and application
- Data characteristics that affect proximity and/or density are
  - Dimensionality and Sparseness
  - Attribute type
  - Special relationships in the data (e.g., autocorrelation)
  - Distribution of the data
- Noise and Outliers
  - Often interfere with the operation of the clustering algorithm

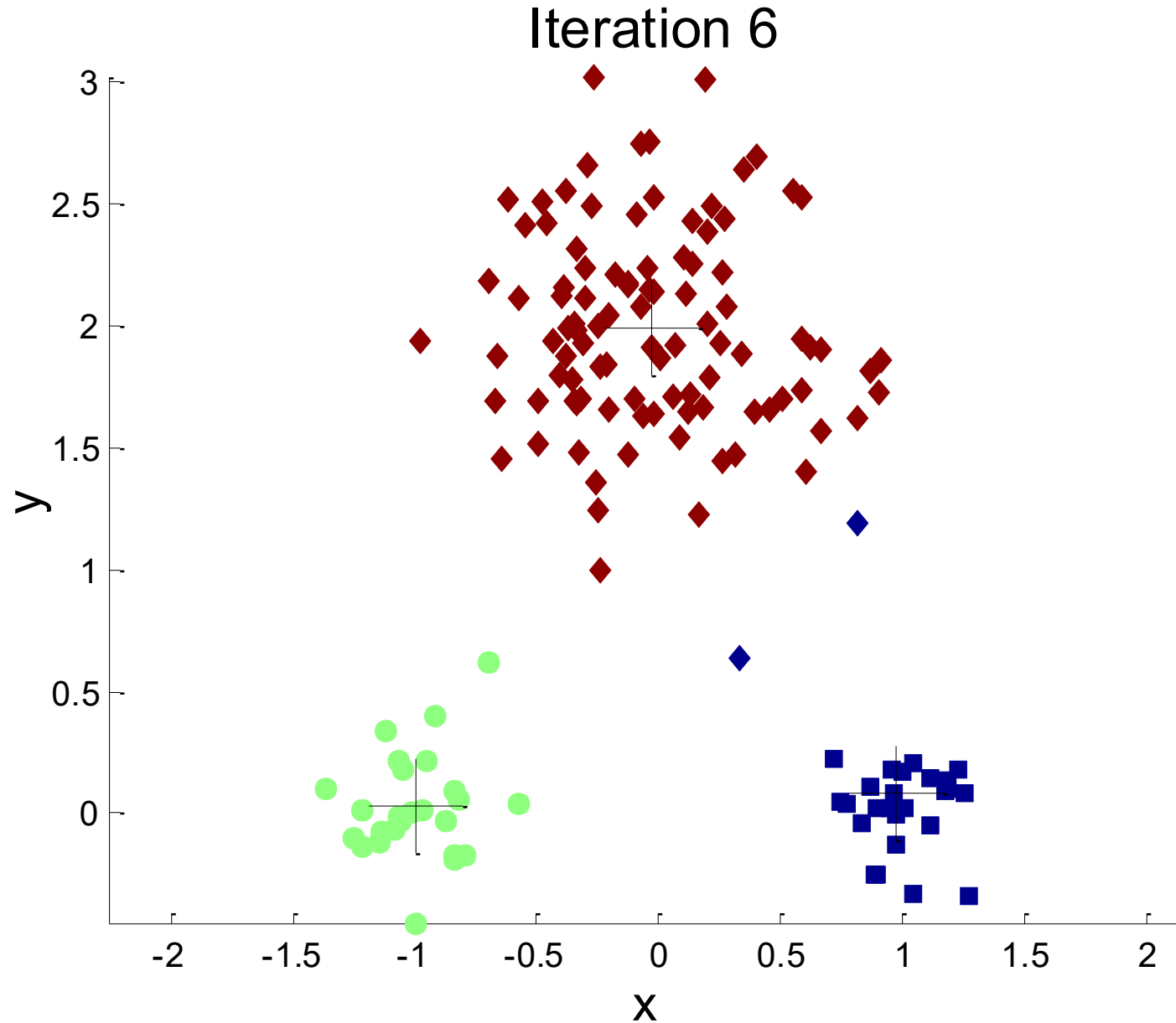
# K-Means Clustering

---

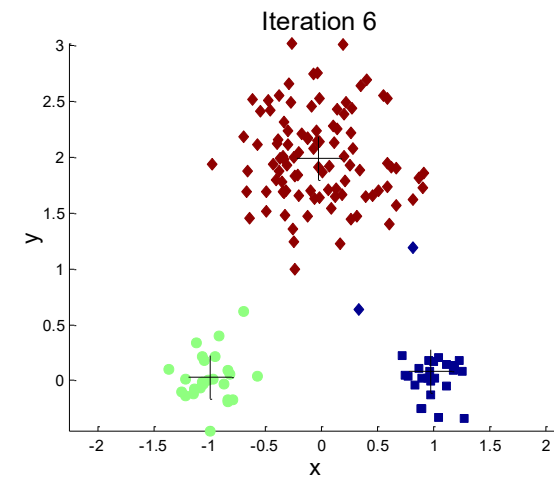
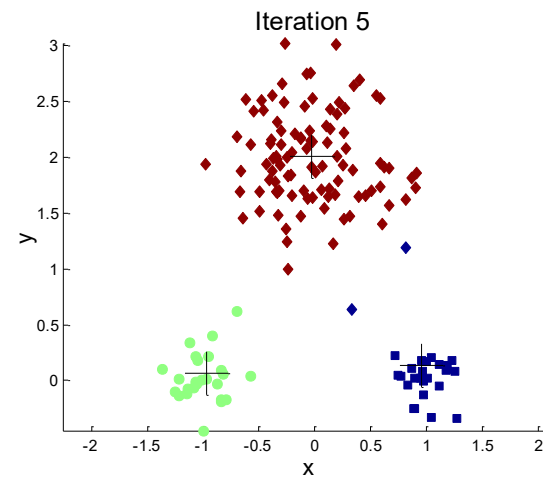
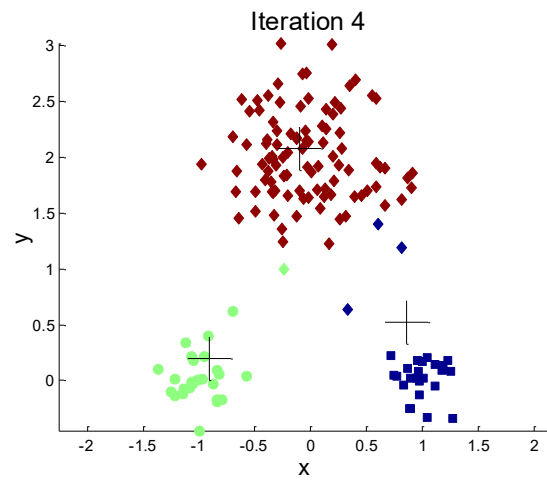
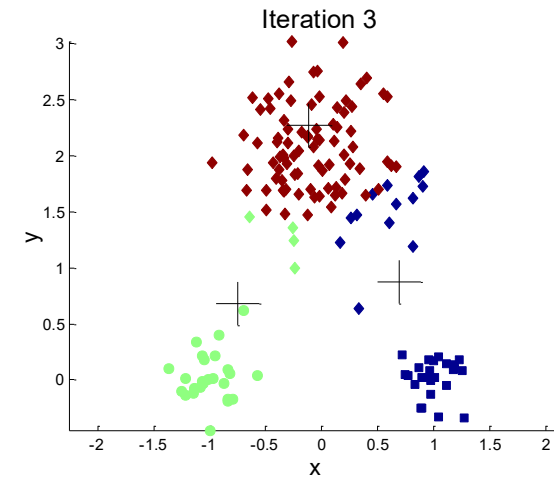
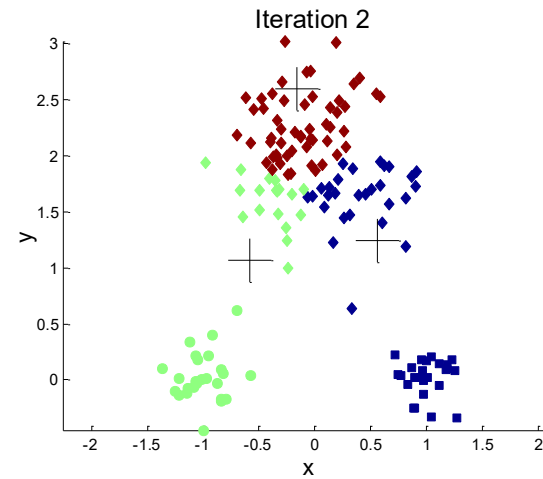
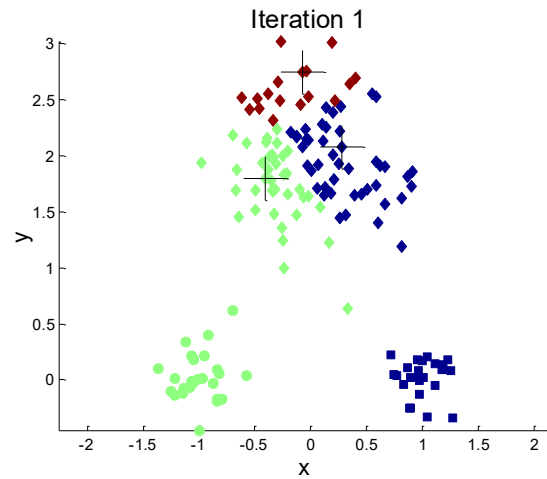
- Partitional clustering approach
- Number of clusters,  $K$ , must be specified
- Each cluster is associated with a **centroid** (center point)
- Each point is assigned to the cluster with the **closest centroid**
- The basic algorithm is very simple

- 
- 1: Select  $K$  points as the initial centroids.
  - 2: **repeat**
  - 3:   Form  $K$  clusters by assigning all points to the closest centroid.
  - 4:   Recompute the centroid of each cluster.
  - 5: **until** The centroids don't change
-

# Example of K-Means Clustering



# Example of K-Means Clustering



# K-Means Clustering – Details

---

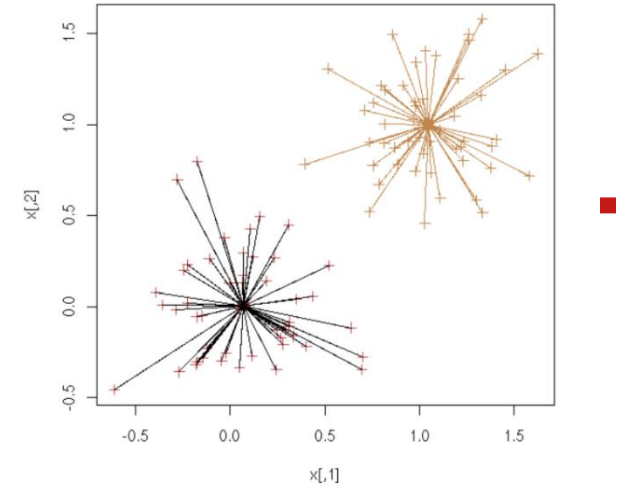
- Initial centroids are often chosen randomly.
  - Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- ‘Closeness’ is measured by Euclidean distance, cosine similarity, correlation, etc.
  - For time series the DTW is usually a solid choice
- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to ‘Until relatively few points change clusters’
- Complexity is  $O(n * K * I * d)$ 
  - $n$  = number of points,  $K$  = number of clusters,  
 $I$  = number of iterations,  $d$  = number of attributes

# Evaluating K-Means Clusters

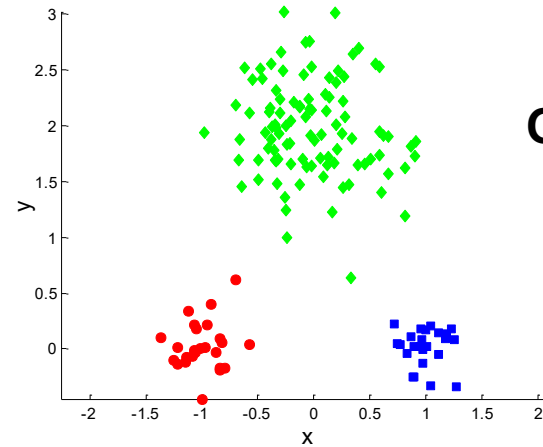
- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster
  - To get SSE, we square these errors and sum them.

$$SSE = \sum_{i=1}^K \sum_{x \in C_i} dist^2(m_i, x)$$

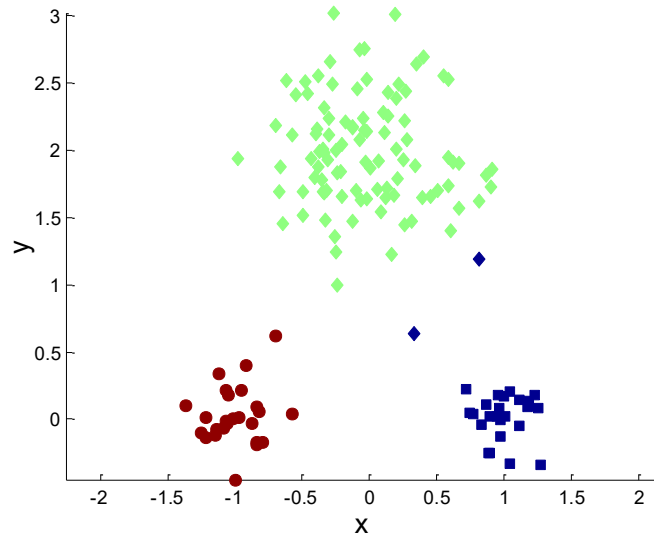
- $x$  is a data point in cluster  $C_i$  and  $m_i$  is the representative point for cluster  $C_i$ 
  - can show that  $m_i$  corresponds to the center (mean) of the cluster
- Given two sets of clusters, we prefer the one with the smallest error
- One easy way to reduce SSE is to increase  $K$ , the number of clusters
- A good clustering with smaller  $K$  can have a lower SSE than a poor clustering with higher  $K$



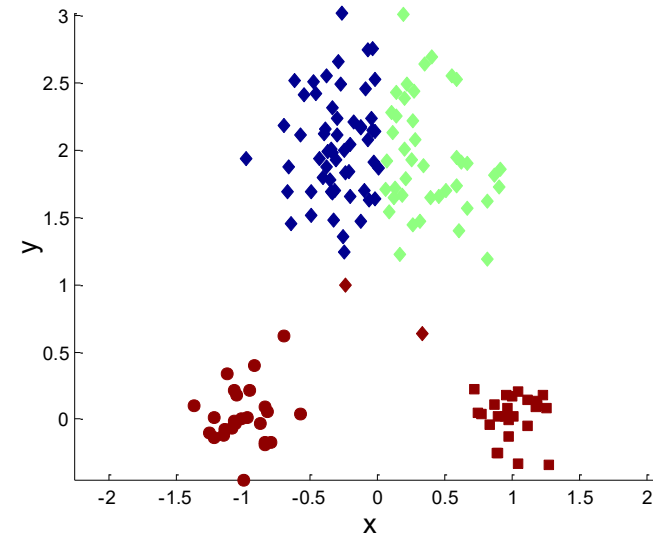
# Two different K-Means Clusterings



Original Points



Optimal Clustering



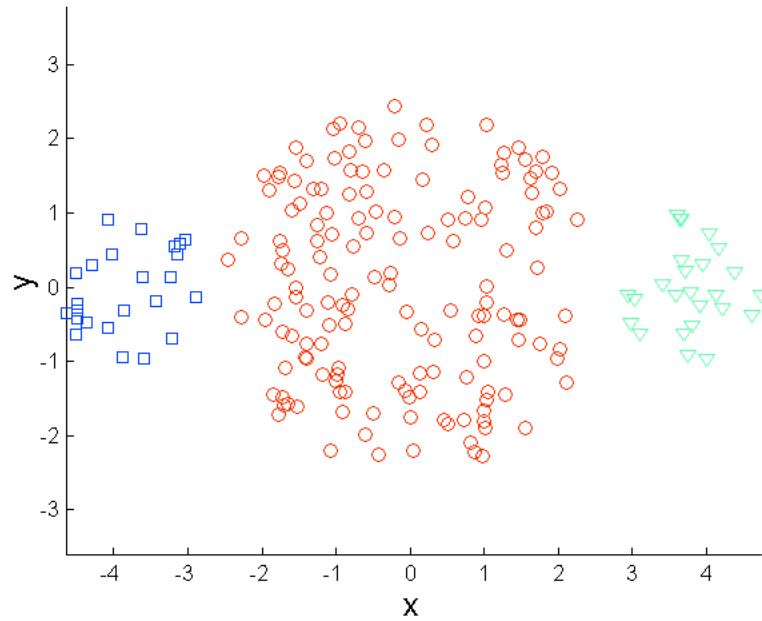
Sub-optimal Clustering

# Limitations of K-Means

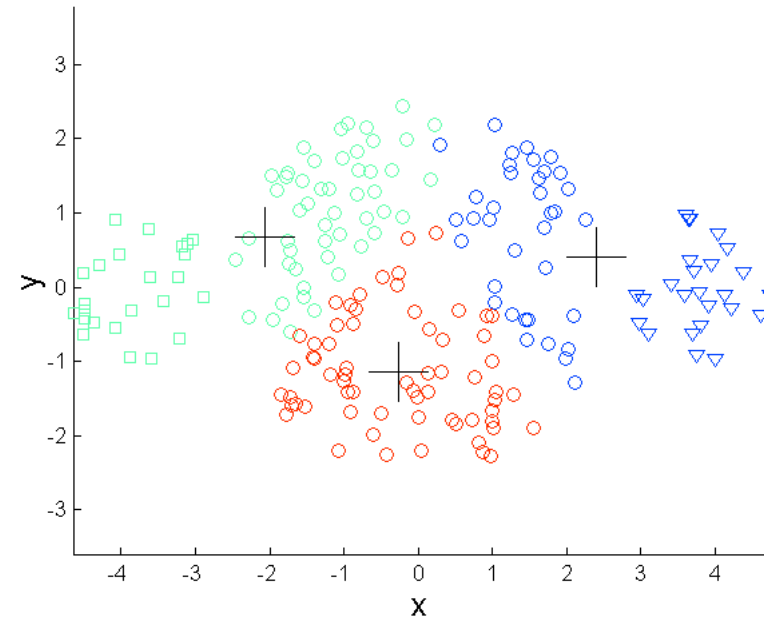
---

- K-Means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes
- K-Means has problems when the data contains outliers.

# Limitations of K-Means: Differing Sizes

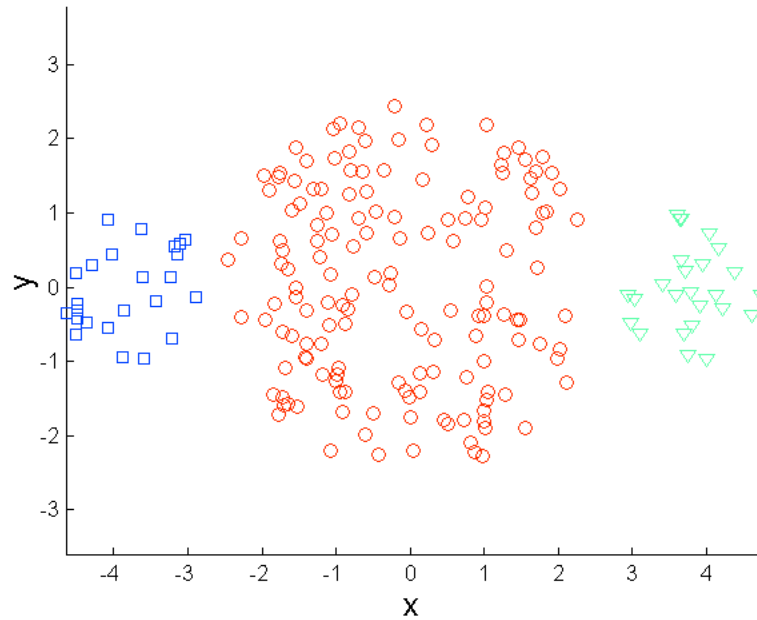


**Original Points**

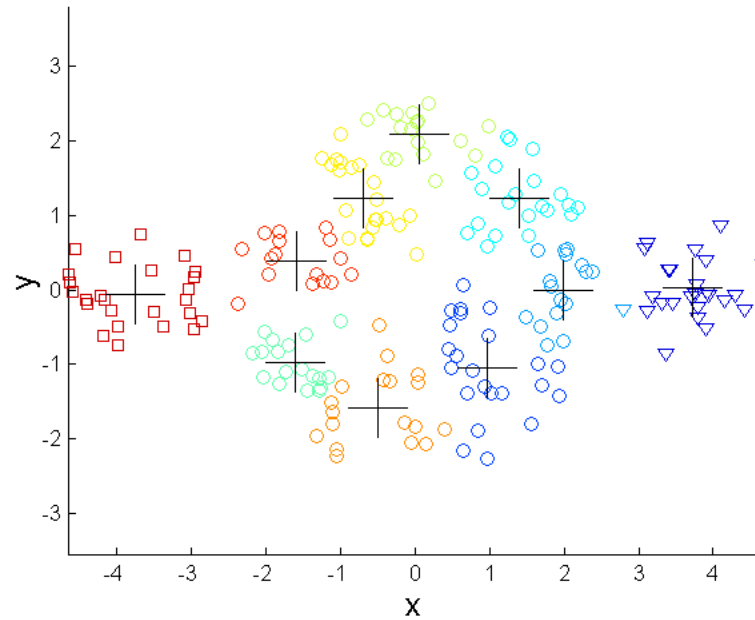


**K-means (3 Clusters)**

# Overcoming K-Means Limitations



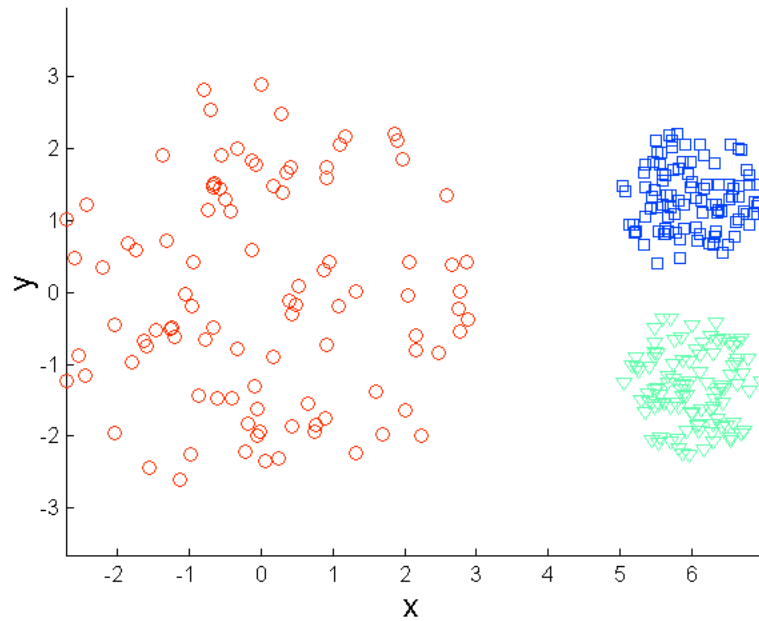
**Original Points**



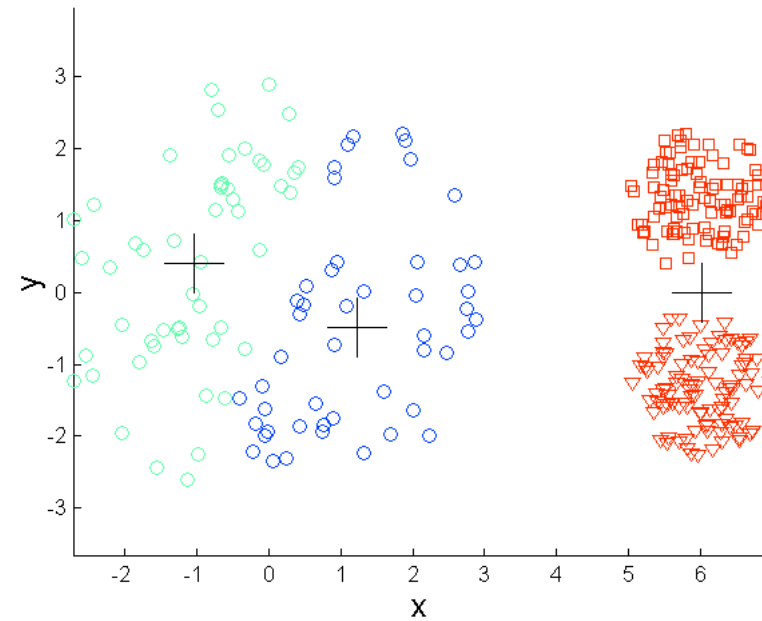
**K-means Clusters**

One solution is to use many clusters.  
Find parts of clusters, but need to put together.

# Limitations of K-Means: Differing Density

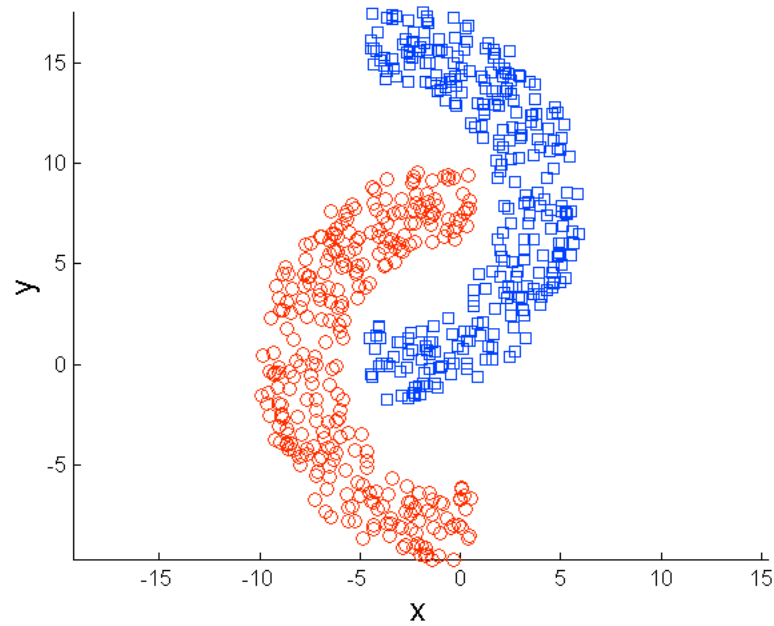


**Original Points**

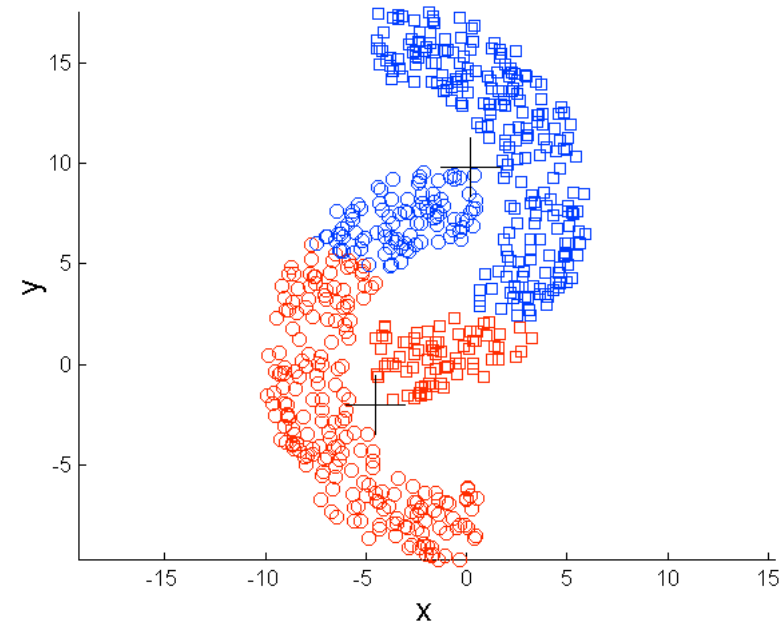


**K-means (3 Clusters)**

# Limitations of K-Means: Non-globular Shapes



**Original Points**



**K-means (2 Clusters)**

# Pre-processing and Post-processing

---

- Pre-processing
  - Normalize the data
  - Eliminate outliers
- Post-processing
  - Eliminate small clusters that may represent outliers
  - Split 'loose' clusters, i.e., clusters with relatively high SSE
  - Merge clusters that are 'close' and that have relatively low SSE
  - Can use these steps during the clustering process
    - ISODATA

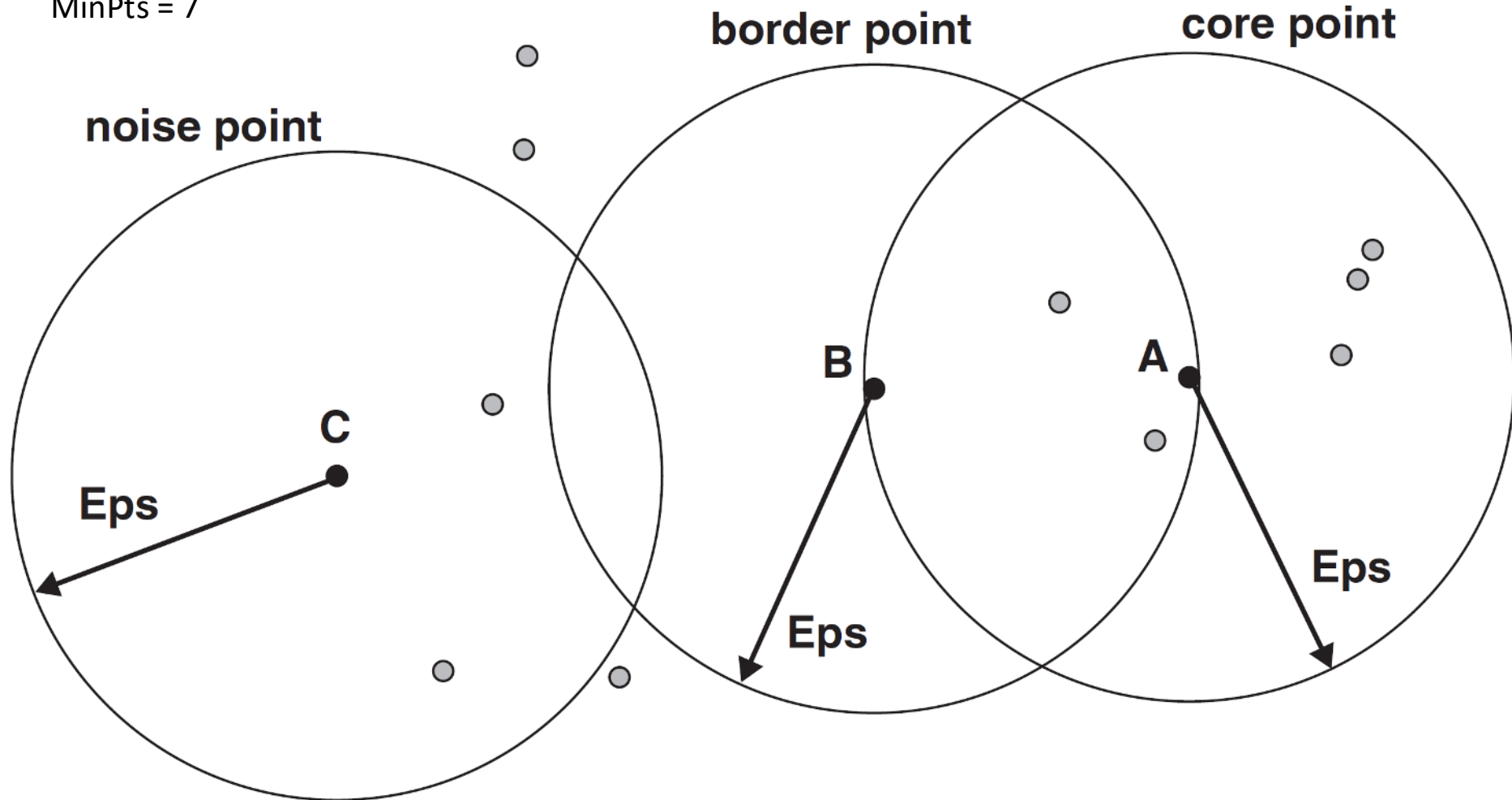
# DBSCAN

---

- DBSCAN is a density-based algorithm.
  - Density = number of points within a specified radius (Eps)
    - The radius is usually in terms of Euclidean distance, but other distances can be used
  - A point is a **core point** if it has at least a specified number of points (MinPts) within Eps
    - These are points that are at the interior of a cluster
    - Counts the point itself
  - A **border point** is not a core point, but is in the neighborhood of a core point
  - A **noise point** is any point that is not a core point or a border point

# DBSCAN: Core, Border, and Noise Points

MinPts = 7



# DBSCAN Algorithm

---

- Eliminate noise points
- Perform clustering on the remaining points

$current\_cluster\_label \leftarrow 1$

**for** all core points **do**

**if** the core point has no cluster label **then**

$current\_cluster\_label \leftarrow current\_cluster\_label + 1$

        Label the current core point with cluster label  $current\_cluster\_label$

**end if**

**for** all points in the  $Eps$ -neighborhood, except  $i^{th}$  the point itself **do**

**if** the point does not have a cluster label **then**

            Label the point with cluster label  $current\_cluster\_label$

**end if**

**end for**

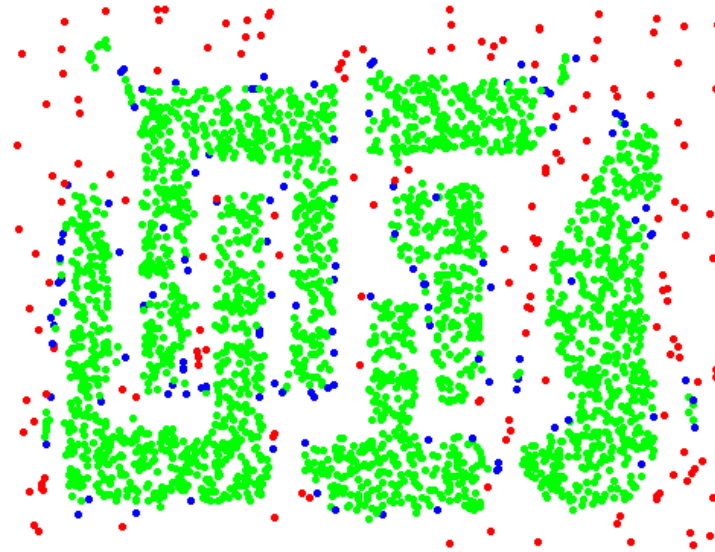
**end for**

# DBSCAN: Core, Border and Noise Points

---



Original Points



Point types: **core**,  
**border** and **noise**

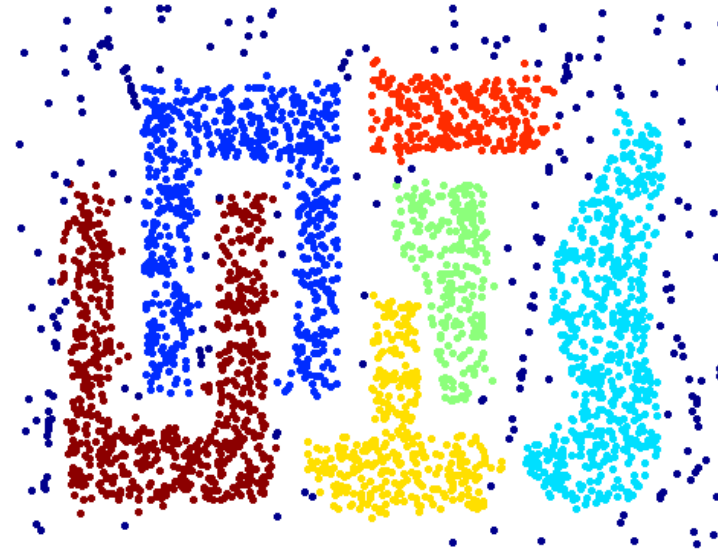
Eps = 10, MinPts = 4

# When DBSCAN Works Well

---



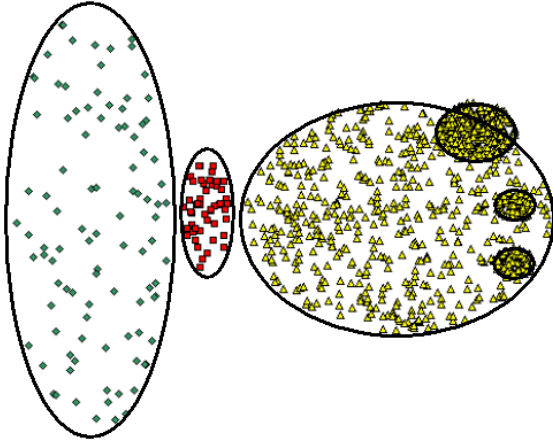
Original Points



Clusters

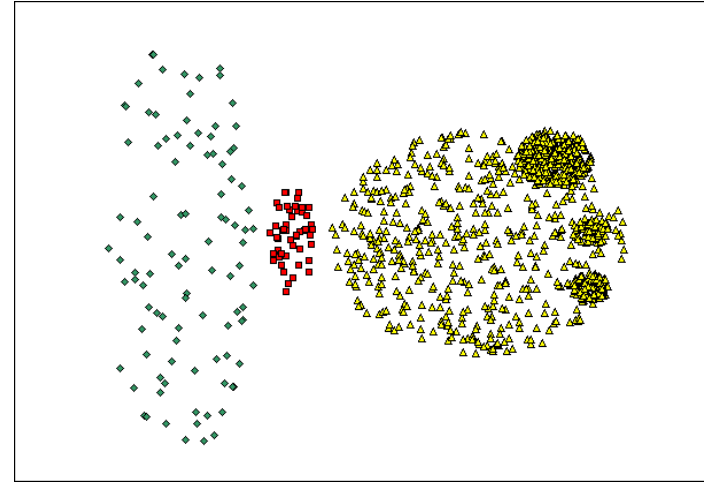
- Resistant to Noise
- Can handle clusters of different shapes and sizes

# When DBSCAN Does NOT Work Well

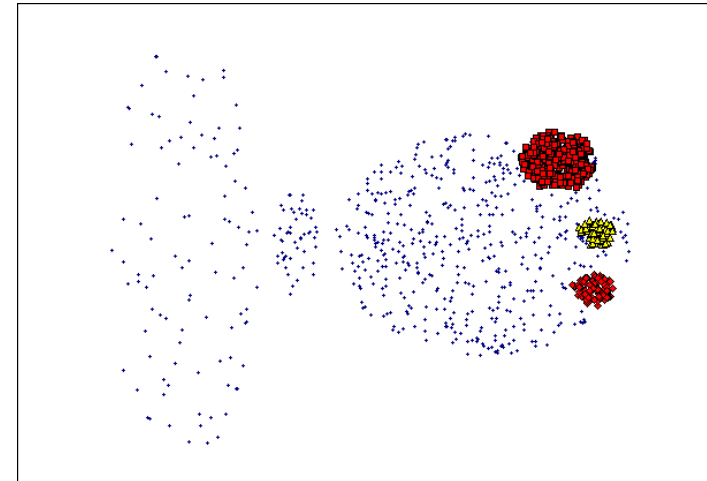


**Original Points**

- **Varying densities**
- **High-dimensional data**



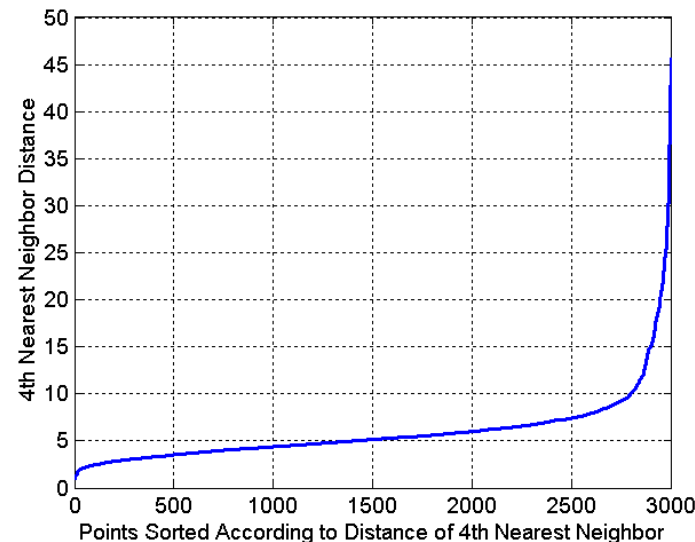
(MinPts=4, Eps=9.75).



(MinPts=4, Eps=9.92)

# DBSCAN: Determining EPS and MinPts

- Idea is that for points in a cluster, their  $k^{\text{th}}$  nearest neighbors are at roughly the same distance
- Noise points have the  $k^{\text{th}}$  nearest neighbor at farther distance
- So, plot sorted distance of every point to its  $k^{\text{th}}$  nearest neighbor



# Time Series Approximations

---

# Dimensionality Reduction

- Dimensionality reduction is the process of reducing the number of variables under consideration by obtaining a subset of principal variables.
- Dimensionality Reduction approaches can be divided into:
  - **Feature Selection:** variables are selected among the existing ones
  - **Feature Projection:** new variables are created to compactly represent the existing ones.

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ | $X_7$ | $X_8$ | $X_9$ | $X_{10}$ |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|----------|
| 1.1   | 1.0   | 0.3   | 0.5   | 0.4   | 1.8   | 1.6   | 1.5   | 1.3   | 2.4      |
| 1.2   | 1.2   | 0.3   | 0.7   | 2.1   | 0.7   | 3.2   | 1.9   | 1.8   | 3.6      |
| ...   | ...   | ...   | ...   | ...   | ...   | ...   | ...   | ...   | ...      |



| $X_2$ | $X_8$ |
|-------|-------|
| 1.0   | 1.5   |
| 1.2   | 1.9   |
| ...   | ...   |

Selection



| $X_A$ | $X_B$ |
|-------|-------|
| 1.8   | 5.4   |
| 1.9   | 6.3   |
| ...   | ...   |

Projection

# Time Series Approximation

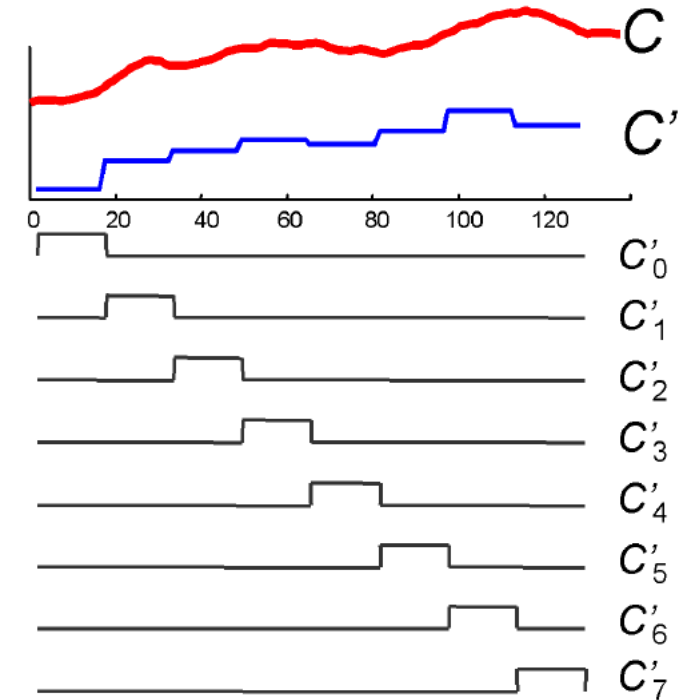
---

- Time Series Approximation is a special form of Dimensionality Reduction specifically designed for TSs.
- Time Series Approximation consists in representing a TS into a smaller and simpler space that is later used for further calculus (e.g. DTW).
- Approximation vs Compression: the approximated space is always understandable, while the compressed space is not necessarily understandable.
- 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)

- PAA approximates a TS by dividing it into equal-length segments and using the mean value of the data points that fall within the segment as representation.
- PPA represent the TS as a sequence of box basis functions with each box of the same size.
- Given  $T = \{x_1, \dots, x_n\}$ , PAA reduces  $T$  from a vector with  $n$  dimensions to a vector  $\bar{T} = \{\bar{x}_1, \dots, \bar{x}_N\}$  with  $N$  dimensions (with  $N < n$ ) by dividing  $T$  into  $N$  equi-sized “frames” with length  $n/N$  where the  $i$ -th element is calculated as

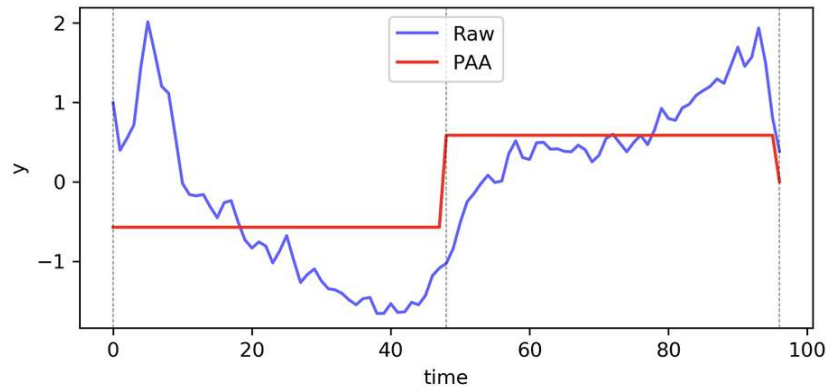
$$\bar{x}_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} x_j$$



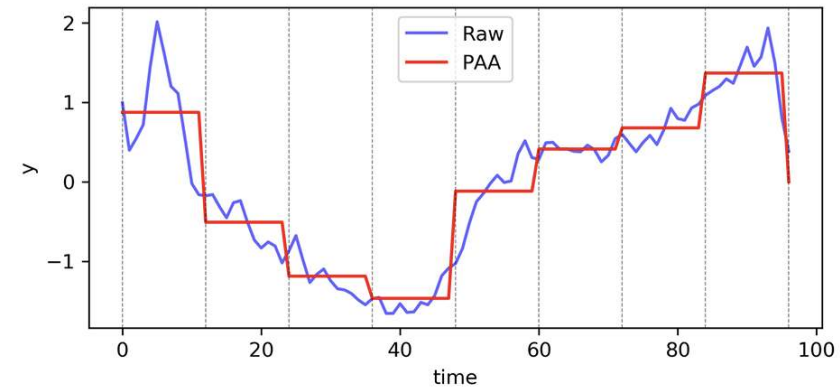
## Pros

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

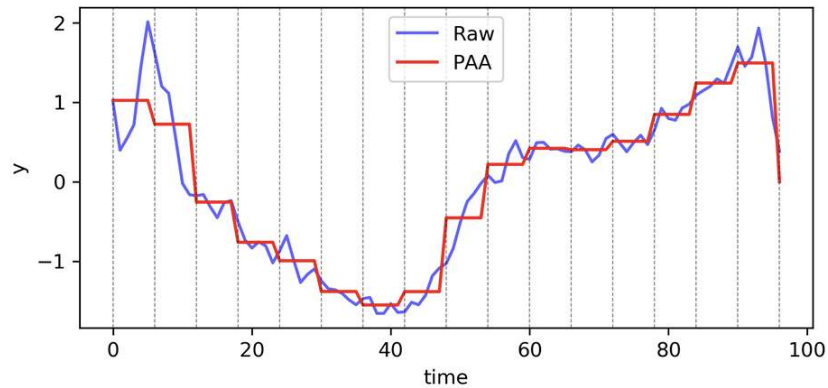
# PAA - Example



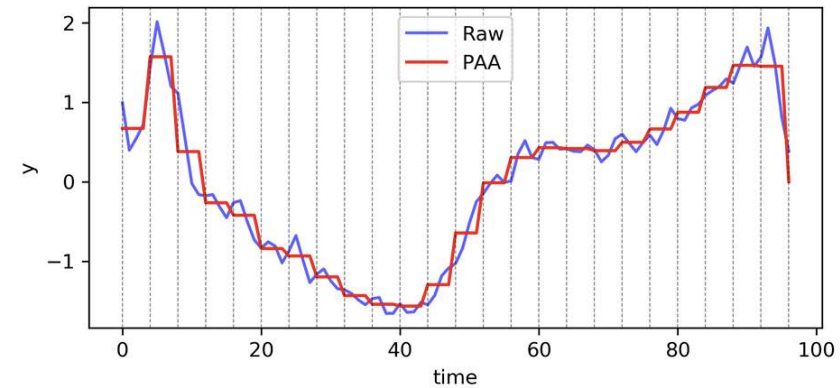
(a) segment  $w = 2$ .



(b) segment  $w = 8$ .

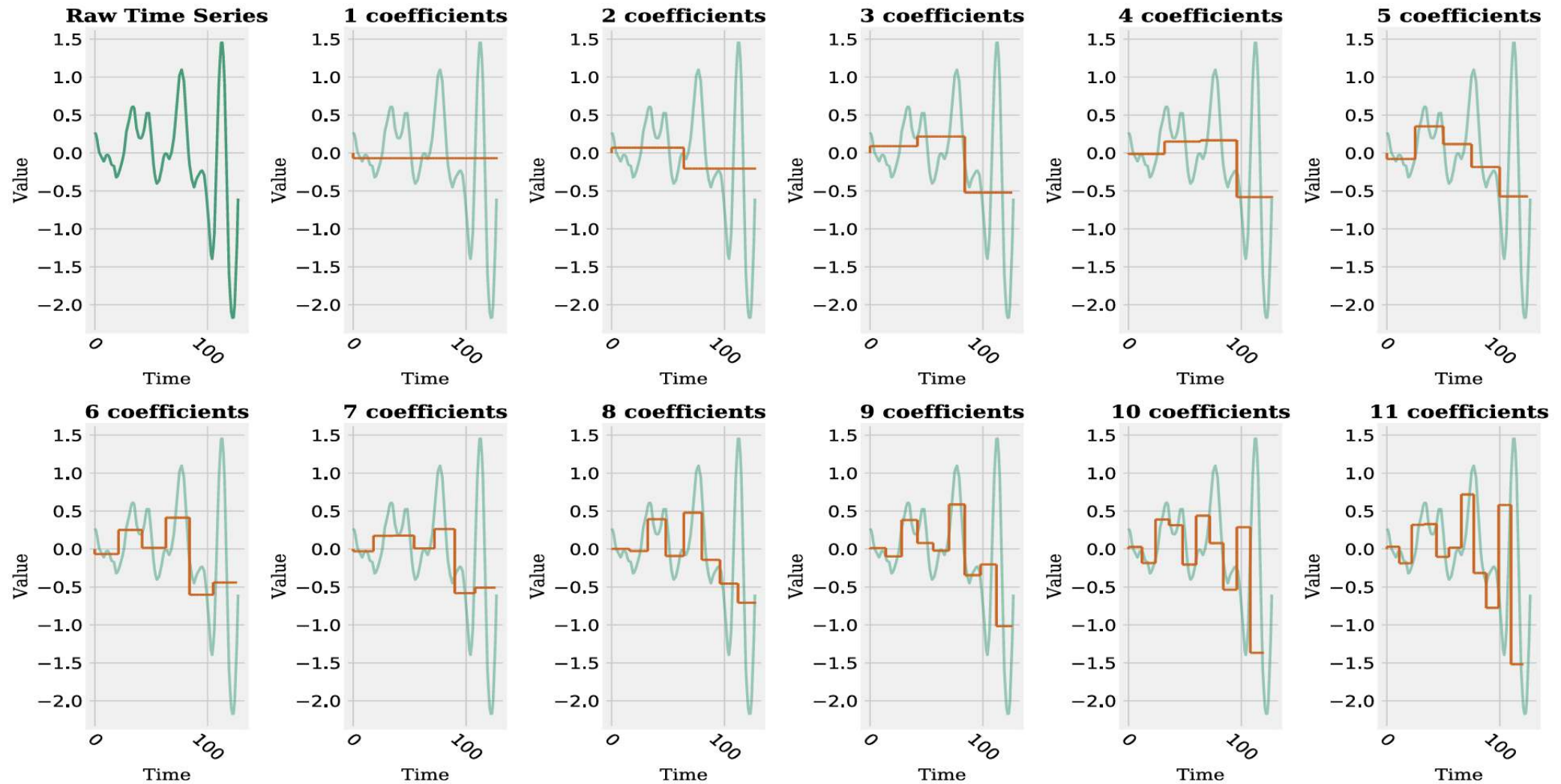


(c) segment  $w = 16$ .



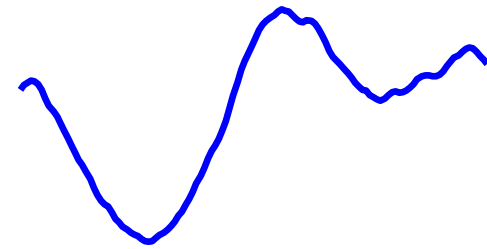
(d) segment  $w = 24$ .

# PAA - Example



# 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.

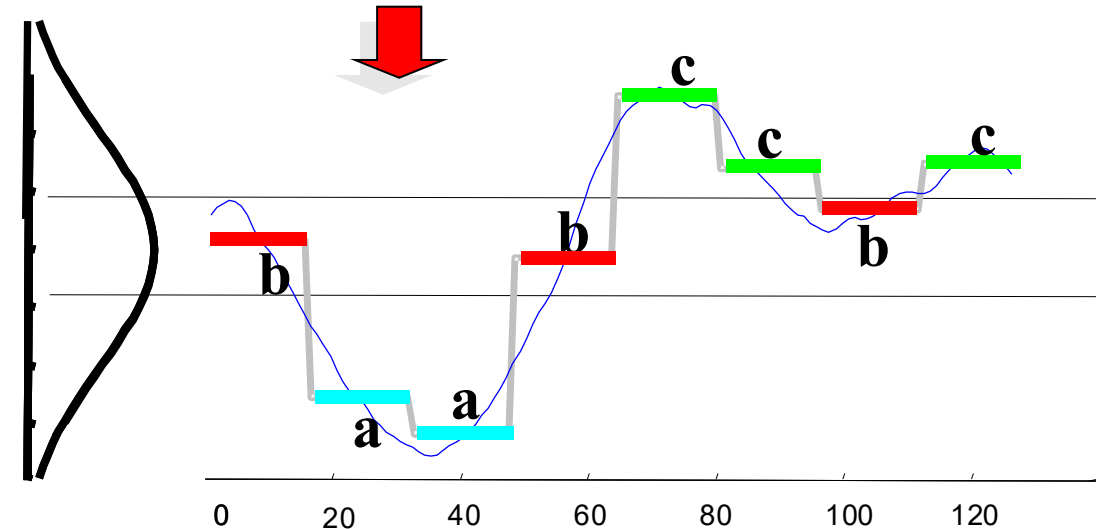
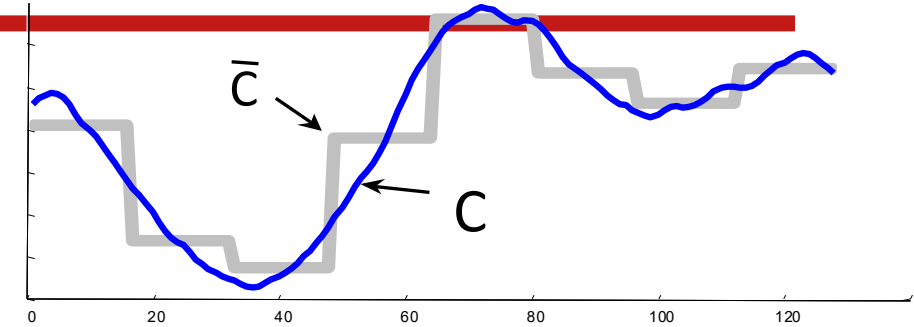


baabccbc



# How do we obtain SAX?

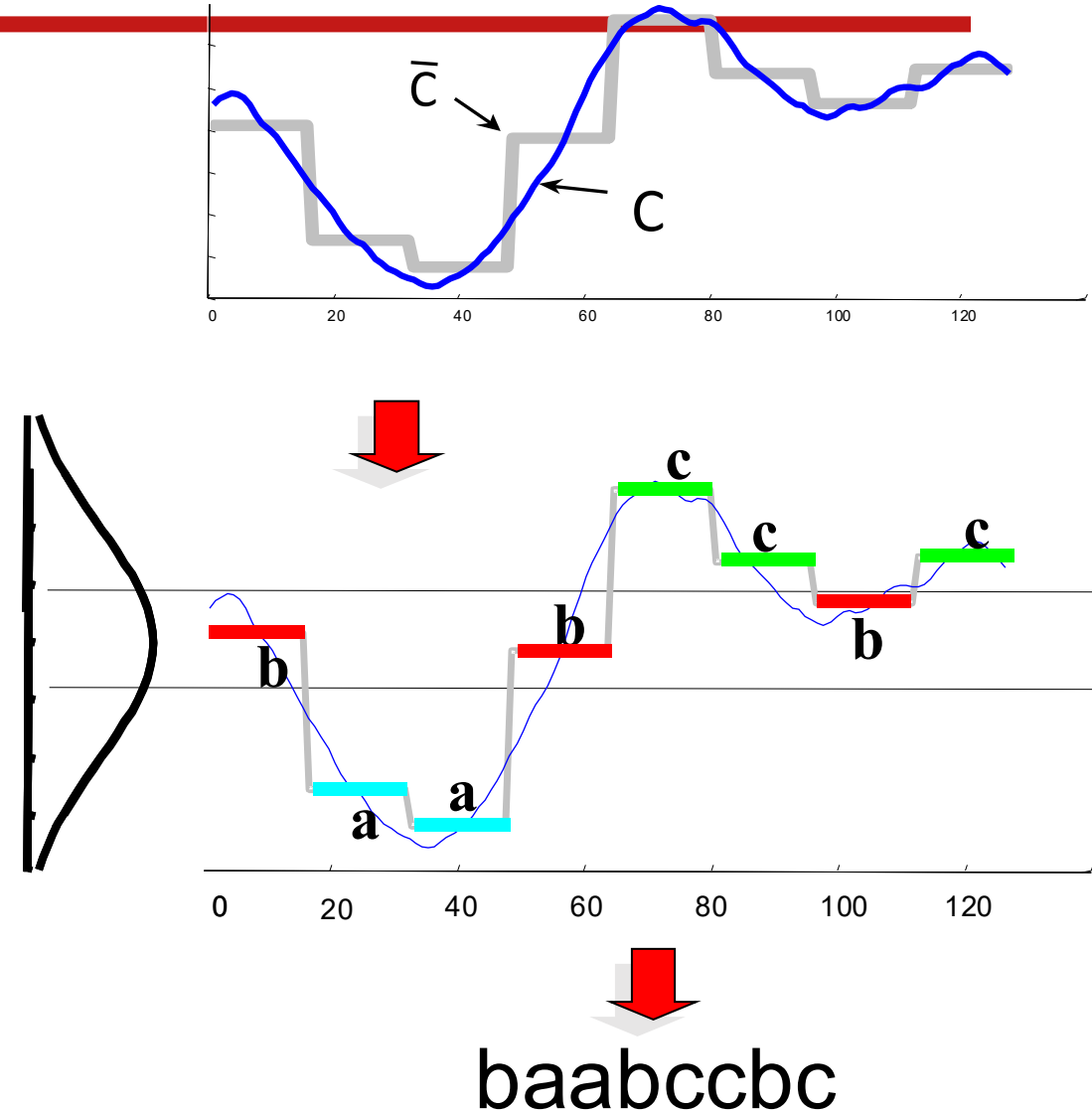
- A time series  $T$  of length  $n$  is divided into  $w$  equal-sized segments; the values in each segment are approximated and replaced by their average.
- Next, we determine the breakpoints that divide the distribution space into  $a$  **equiprobable** regions, where  $a$  is the alphabet size specified by the user.
- The breakpoints are determined such that the probability of a segment falling into any of the regions is approximately the same.



baabccbc

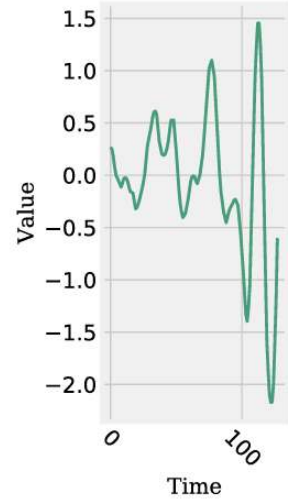
# How do we obtain SAX?

- Once the breakpoints are determined, each region is assigned to a symbol and the PAA coefficients are mapped with the symbol corresponding to the region 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.

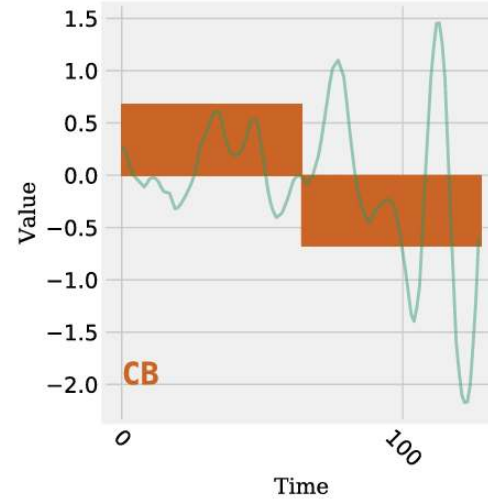


# SAX - Example

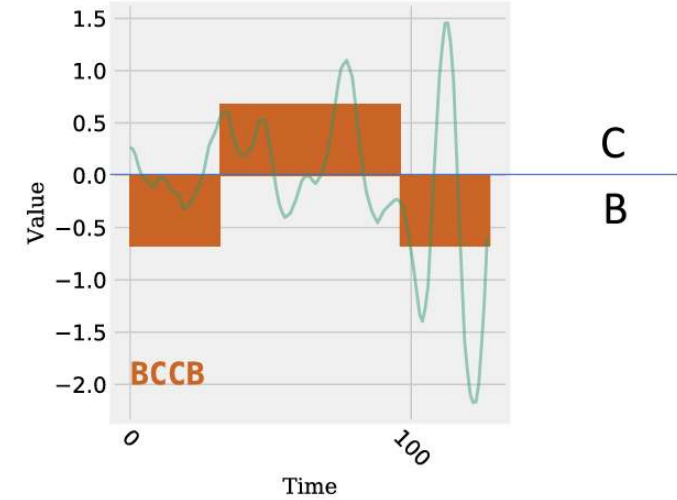
Raw Time Series



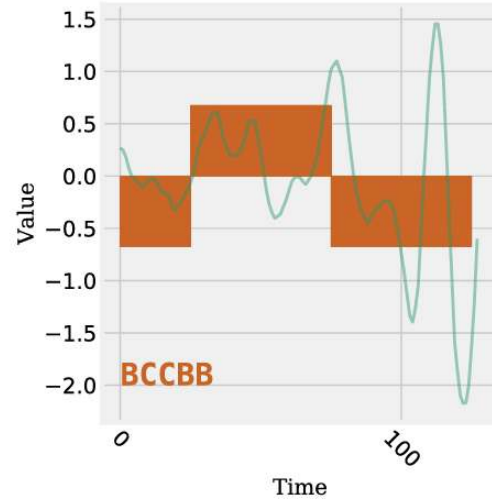
2 coefficients



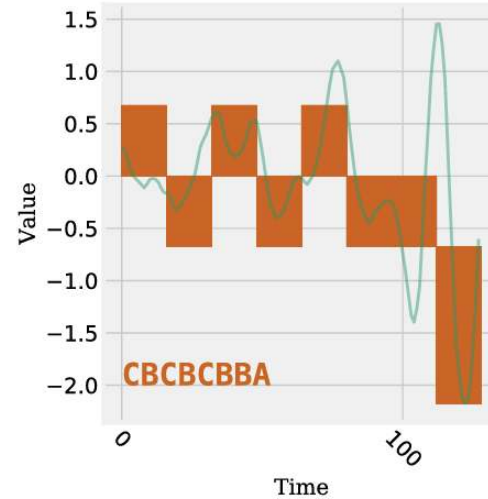
4 coefficients



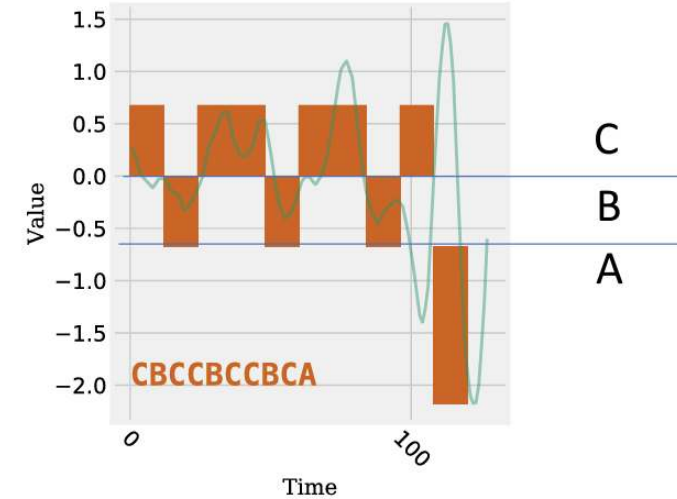
5 coefficients



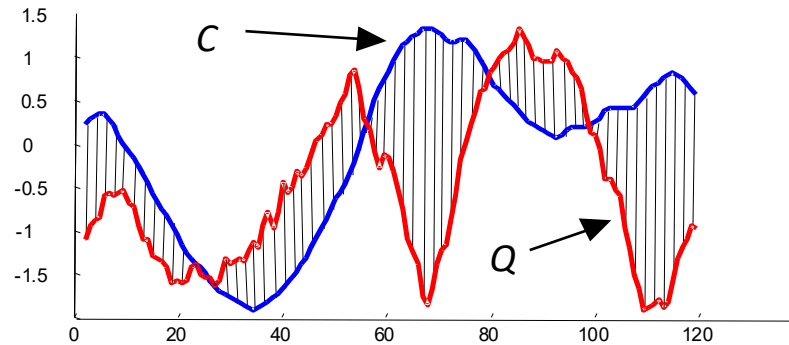
8 coefficients



10 coefficients



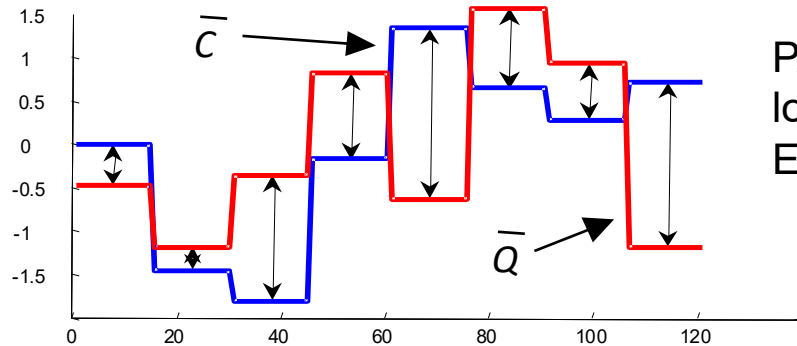
# Distances and Approximations



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

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

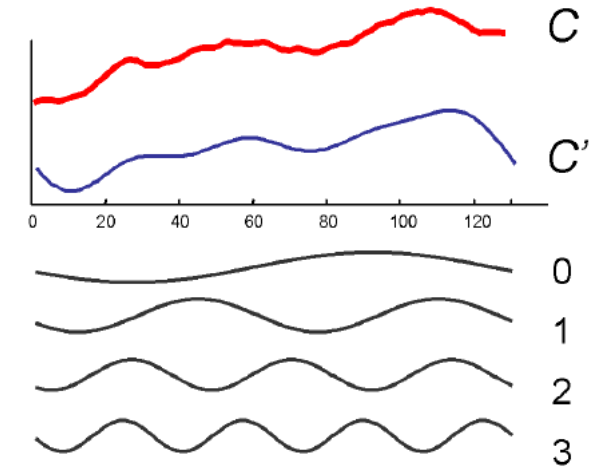
$\hat{C}$  = baabccbc  
 $\hat{Q}$  = babcacca

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

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

# Discrete Fourier Transform (DFT)

- Represent a TS of length  $n$  as a linear combination of  $w$  smooth periodic sinusoidal series (i.e., sines and cosines).
- Each wave is represented by a Fourier coefficient
- This representation is called Frequency Domain
- The DFT concentrates most of its energy in the first few Fourier coefficients
- Low-pass filter: approximate a TS by its first  $w$  Fourier coefficients.



# How do we obtain DFT?

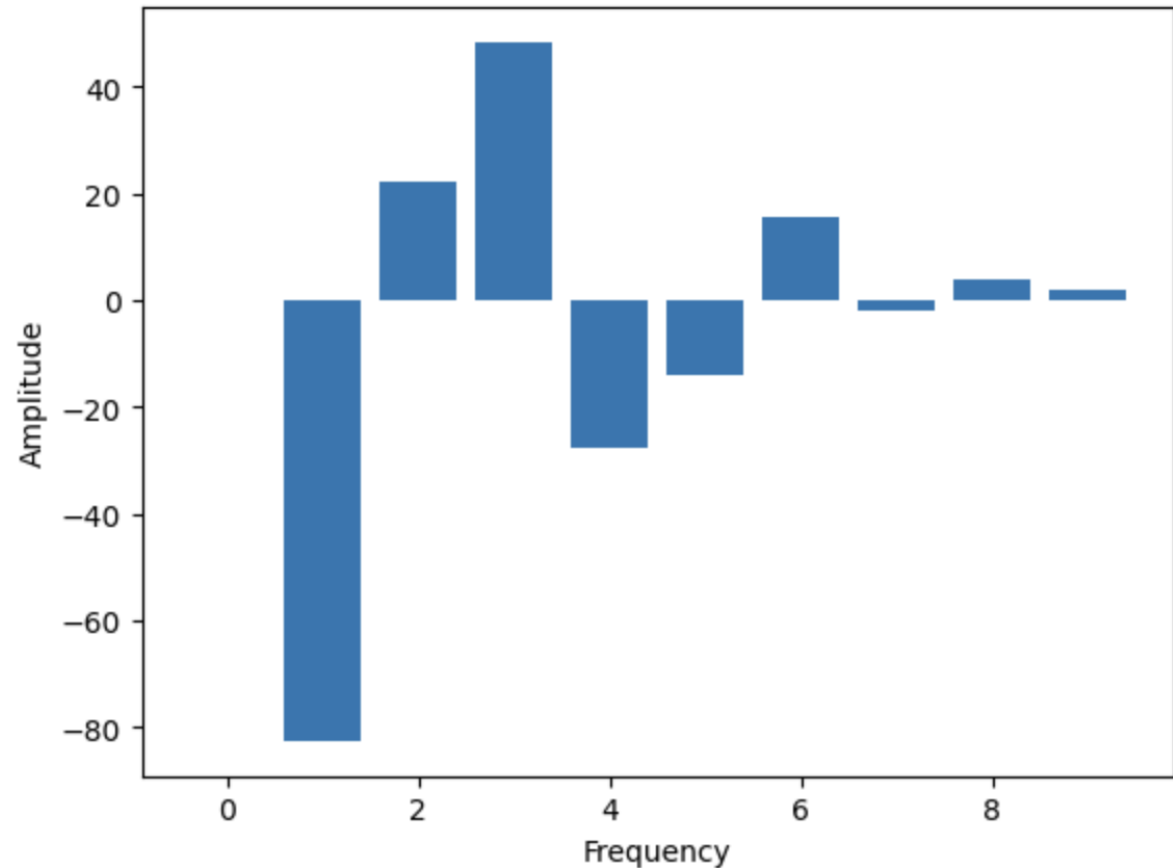
---

- The DFT decomposes a TS  $T$  of length  $n$  into a sum of  $n$  orthogonal basis functions using sinusoid waves represented with two numbers: amplitude and phase.
- A Fourier coefficient (sinusoid wave) is represented by the complex number:  $X_u = (real_u, imag_u)$   $u = 0, 1, \dots, n-1$
- The  $n$ -th point DFT of a TS  $T = \{x_1, \dots, x_n\}$  is given by:  $DFT(T) = X_0, \dots, X_{n-1}$
- with  $X_u = \frac{1}{n} \sum_{i=1}^n x_i e^{-\frac{j2\pi i u}{n}}$   $u \in [0, n), j = \sqrt{-1}$
- The first Fourier coefficient ( $X_0 = \frac{1}{n} \sum_{i=1}^n x_i e^0$ ) is equal to the mean of a TS and can be discarded to obtain offset invariance, i.e., level removal.

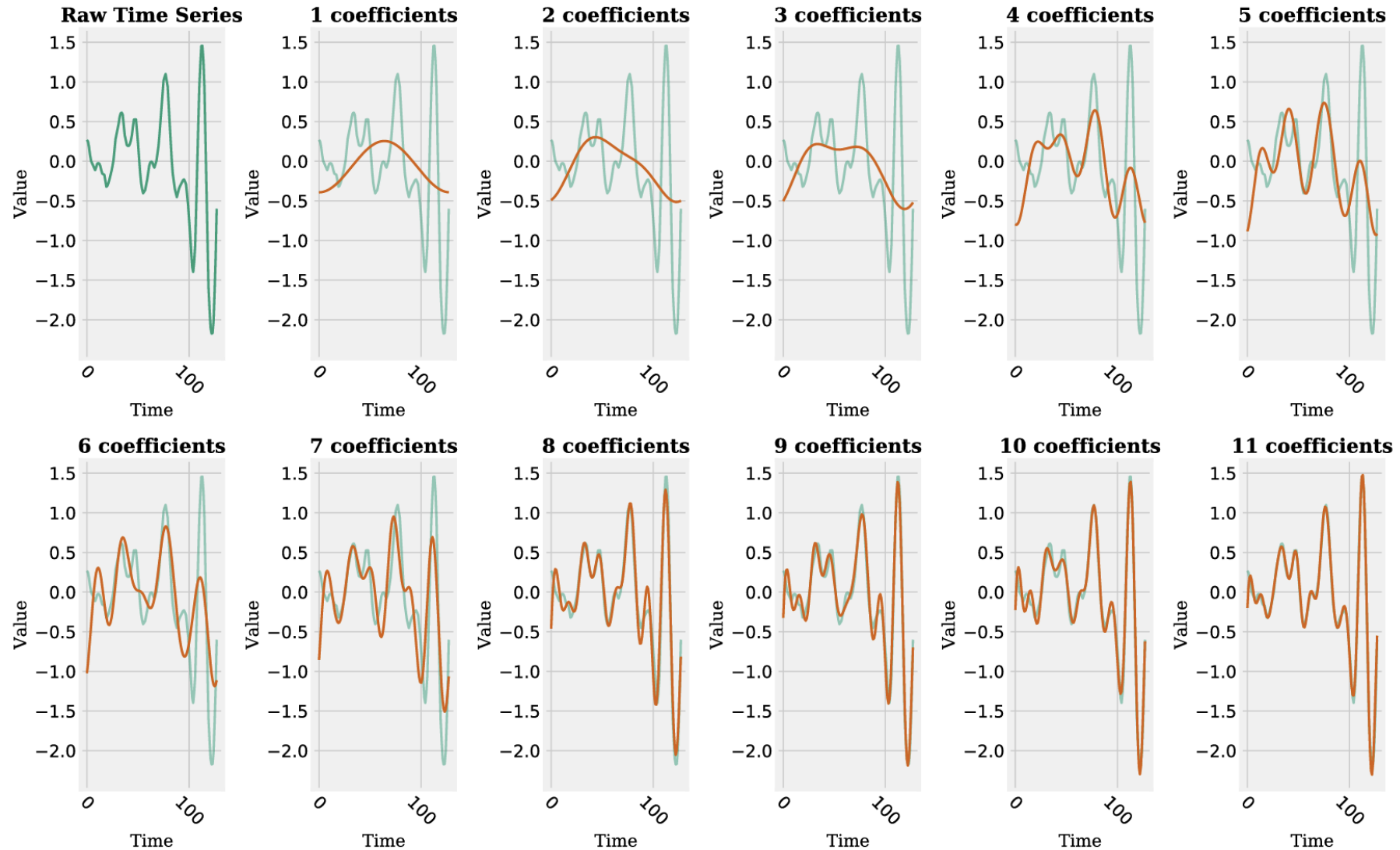
# How do we obtain DFT?

- You usually do not plot  $X_u$  directly as a single real-valued y-axis.
- Most commonly, you plot **magnitude**, also called amplitude:

$$|X_u| = \sqrt{\text{real}_u^2 + \text{imag}_u^2}$$

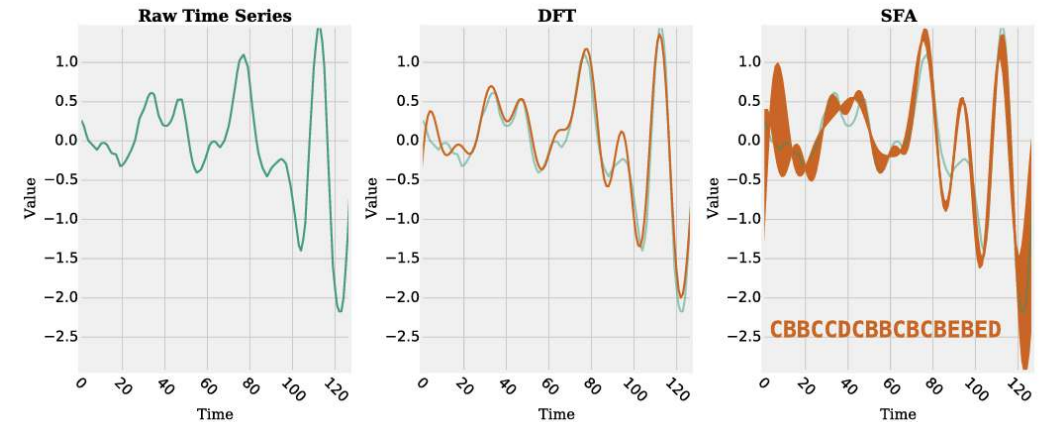


# DFT - Example



# Symbolic Fourier Approximation (SFA)

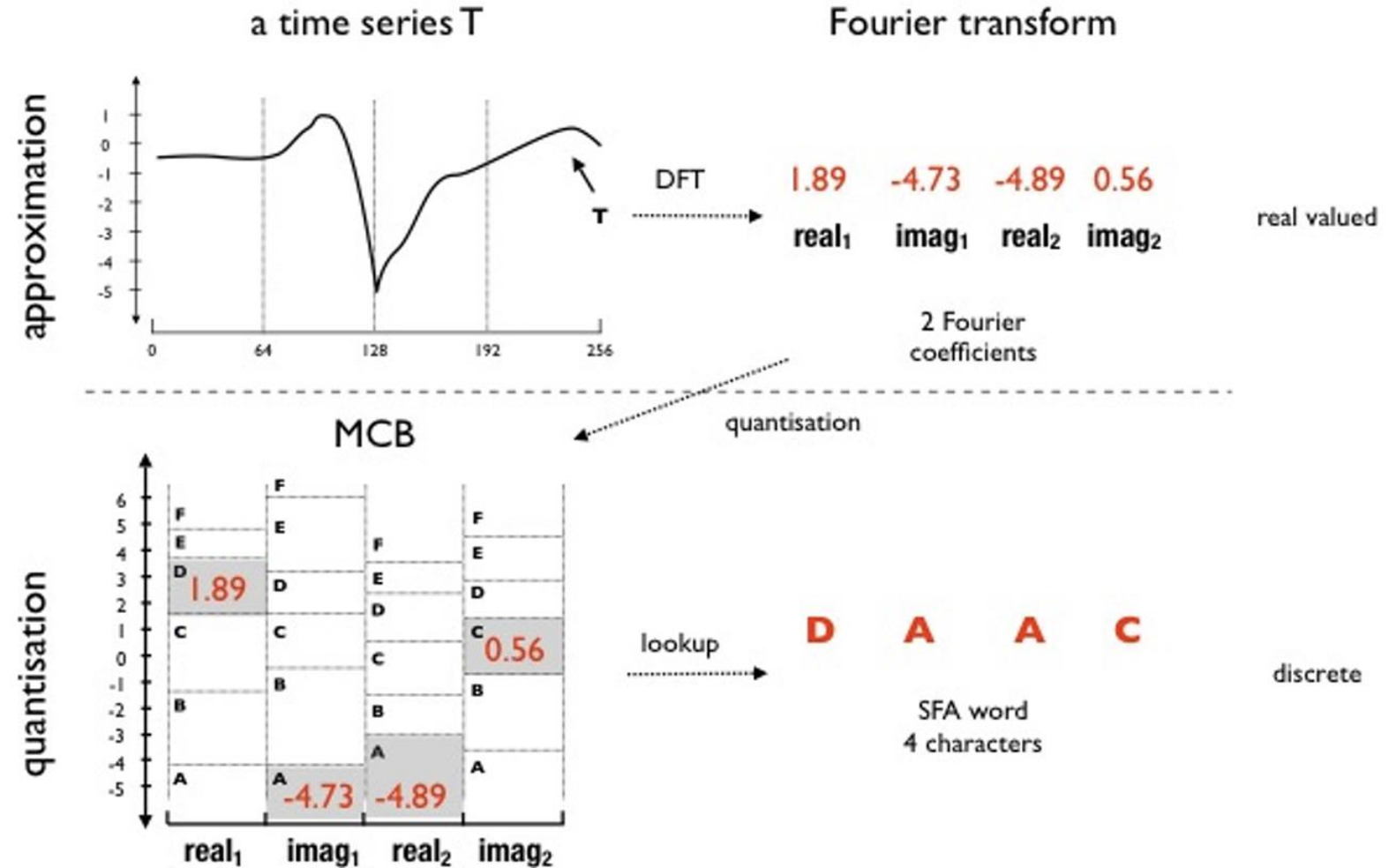
- SFA represents a TS with a word
- SFA is composed of
  - a) approximation using the Fourier transform
  - b) a data adaptive discretization
- The **discretization intervals are learned** from the Fourier transformed data distribution rather than using fixed intervals with Multiple Coefficient Binning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

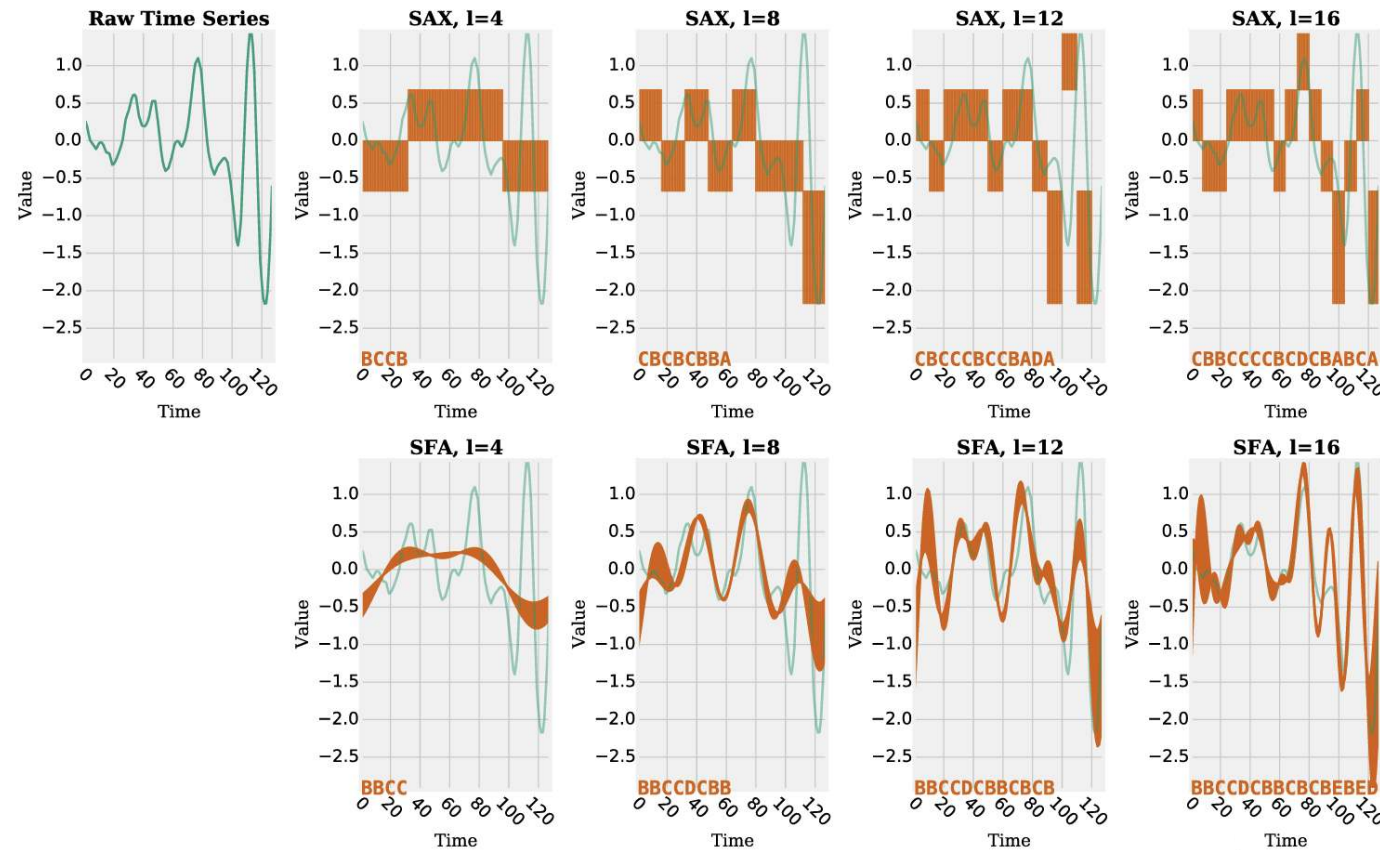
# SFA Discretization

- 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
  - SAX = PAA + Discretization
  - SFA = DFT + 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.



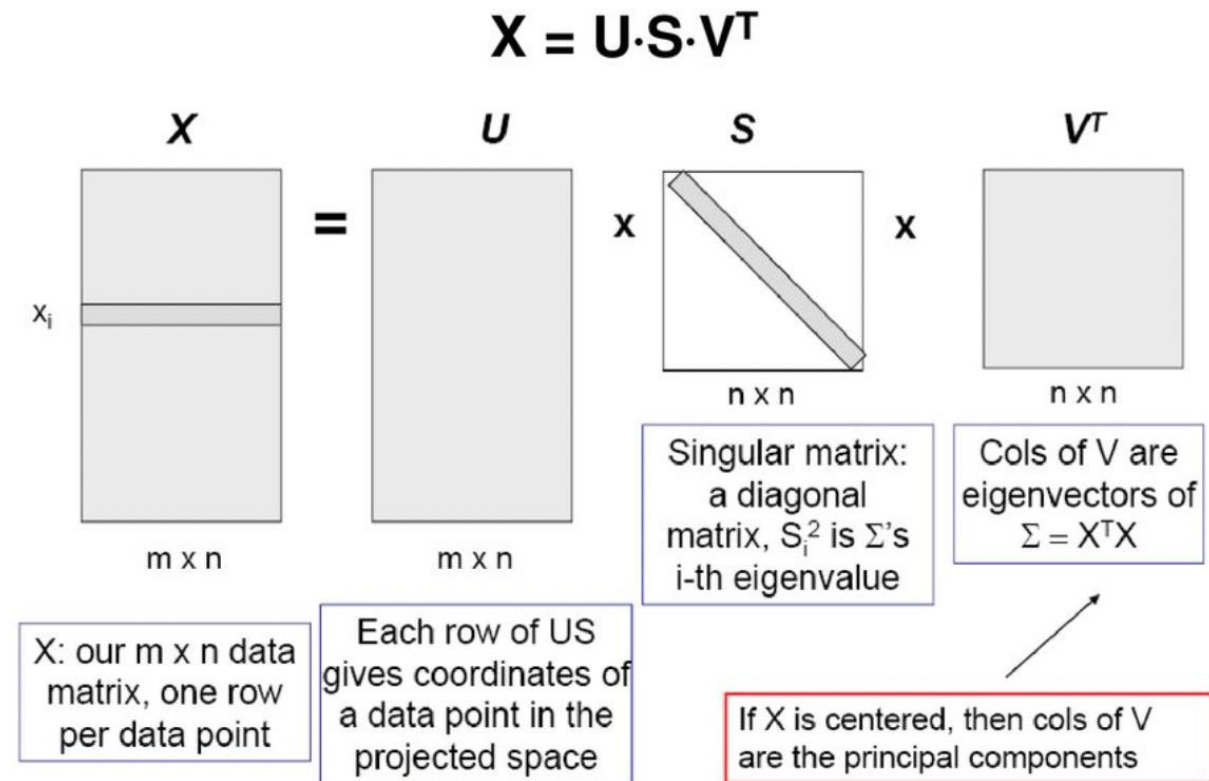
# Properties of Symbolic Representations

---

- 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

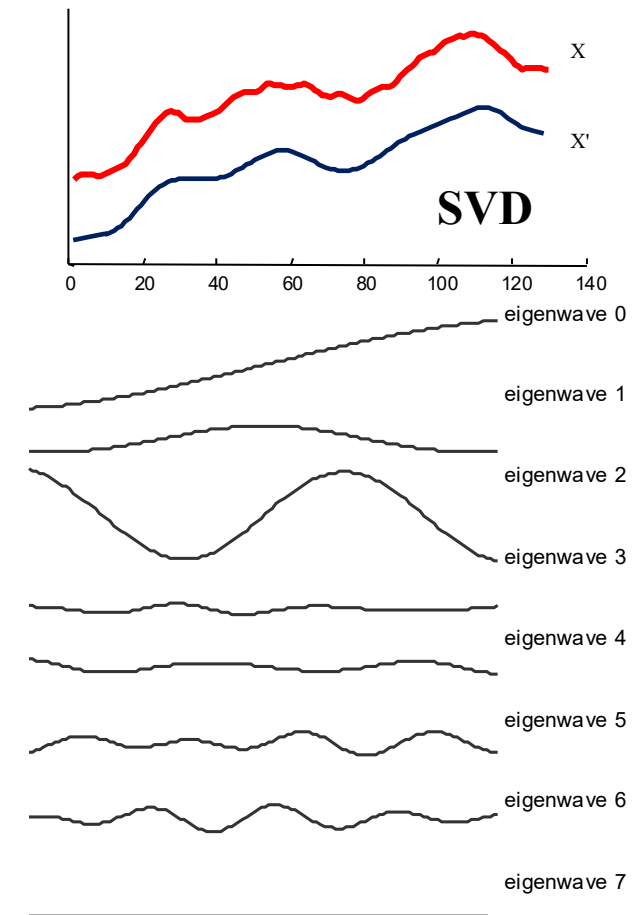
# Singular Value Decomposition (SVD)

- 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.



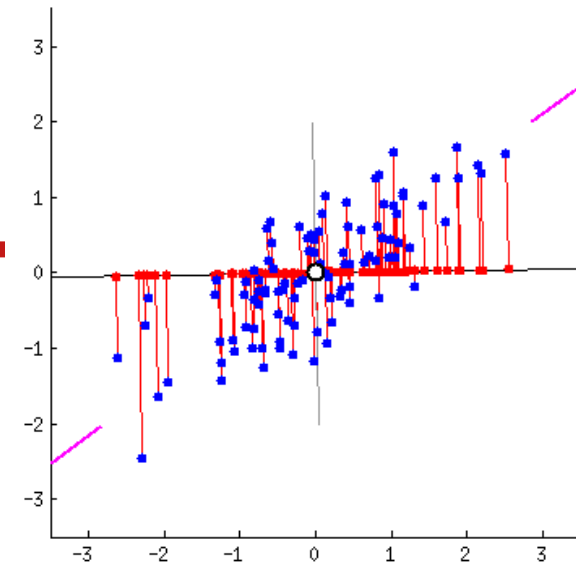
# SVD for Time Series Approximations

- SVD is similar to DFT, it represents the shape in terms of a linear combination of basis shapes.
- DFT is an **individual approximations** as it examines one TS at a time. It is 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  $\Sigma$ .
  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

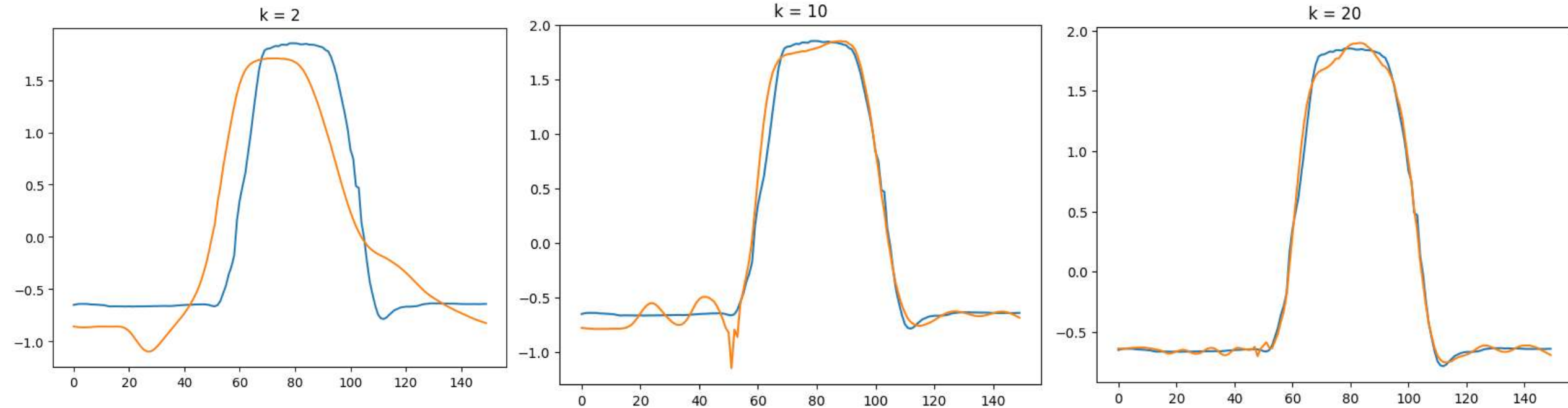
- Both PCA and SVD involve eigen-decomposition.
- PCA performs eigen-decomposition on the covariance matrix of the data which is a square symmetric matrix of size  $m \times m$  where  $m$  is the number of time stamp.
- SVD performs eigen-decomposition on the data matrix itself of size  $n \times m$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



# Remarks on Approximations

---

- None of the representations can be superior for all tasks.
- No research as 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

---

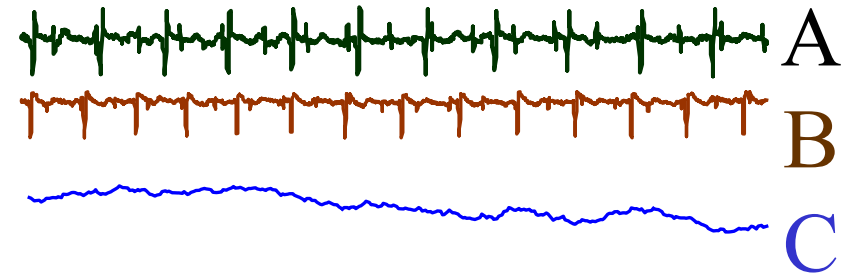
- It does not make any sense to use a distance function accounting for time like DTW if a Time-Independent approximation is used.
- Thus, do not use DTW after DTF, SFA, SVD, PCA.
- It does make sense to use DTW after Time-Dependent approximations.
- Thus, you can use DTW after PAA or SAX.
- Normalizations 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.

# Global Structural Features

---

# Structure-based Similarity

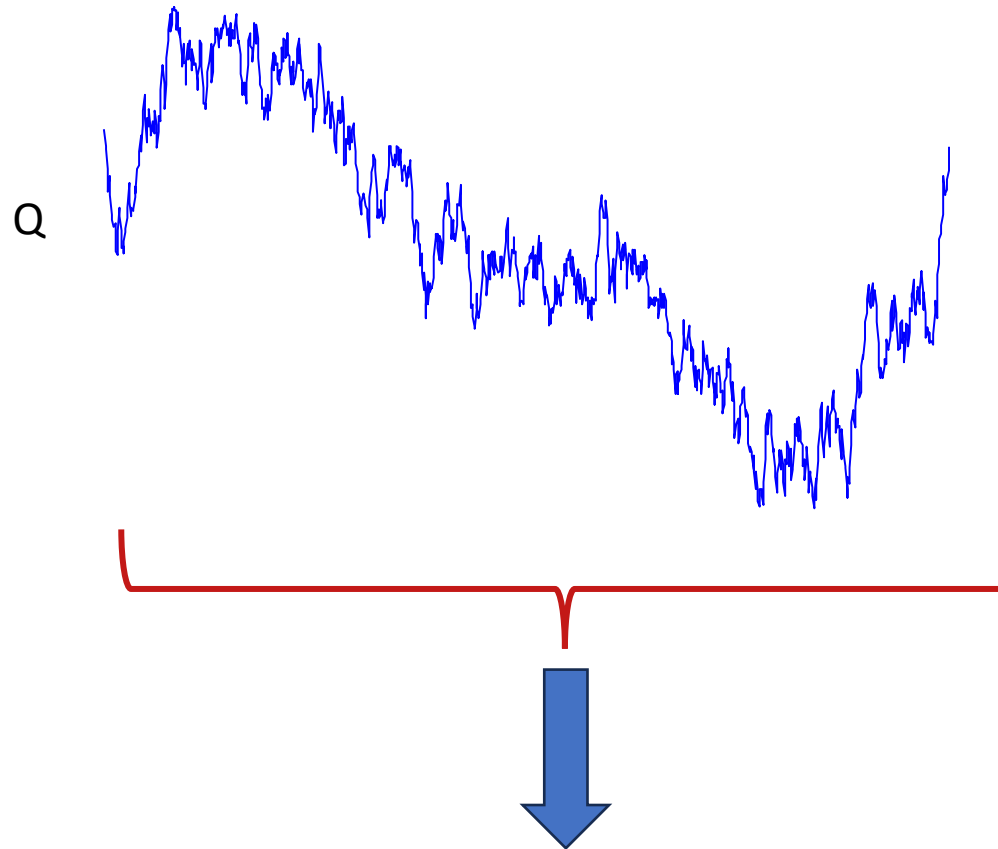
- For long time series, shape-based similarity typically give poor results.
- Structure-based similarity measure similarity of TS based on high level structure.
- The basic idea is to:
  1. extract **global** features from the time series,
  2. create a feature vector, and
  3. use it to measure similarity with Euclidean distance
- Example of features:
  - mean, variance, skewness, kurtosis,
  - 1<sup>st</sup> derivative mean, 1<sup>st</sup> 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 |
| ...                 | ... | ... | ... |

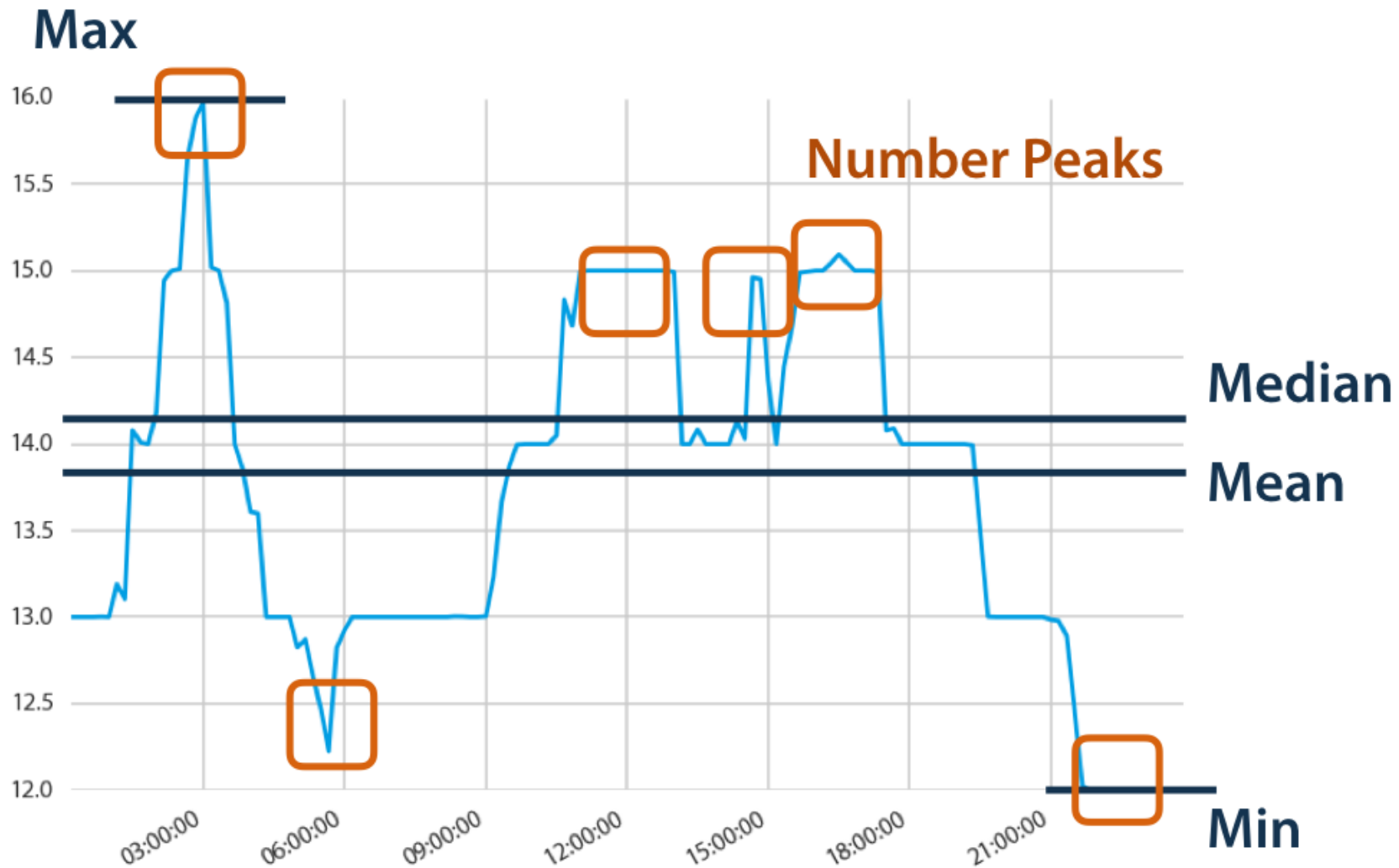
# Simple Standard Features

- Mean
- Standard Deviation
- Variance
- Median
- 10th Percentile
- 25th Percentile
- 75th Percentile
- 90th Percentile
- IQR
- Covariance
- Skewness
- Kurtosis
- Min
- Max



|   | mean | std | var | med | ... | max |
|---|------|-----|-----|-----|-----|-----|
| Q | 1.8  | 2.9 | 8.4 | 1.3 | ... |     |

# TSFresh Features



# TSFresh Features

- `abs_energy` - Returns the absolute energy of the time series which is the sum over the squared values
- `absolute_maximum` - Calculates the highest absolute value of the time series `x`.
- `absolute_sum_of_changes` - Returns the sum over the absolute value of consecutive changes in the series `x`
- `agg_autocorrelation` - Descriptive statistics on the autocorrelation of the time series.
- `agg_linear_trend` - Calculates a linear least-squares regression for values of the time series that were aggregated over chunks versus the sequence from 0 up to the number of chunks minus one.
- `approximate_entropy` - Implements a vectorized Approximate entropy algorithm.
- `ar_coefficient` - This feature calculator fits the unconditional maximum likelihood of an autoregressive AR(k) process.
- `augmented_dickey_fuller` - Does the time series have a unit root?
- `autocorrelation` - Calculates the autocorrelation of the specified lag
- `benford_correlation` - Useful for anomaly detection applications. Returns the correlation from first digit distribution when
- `binned_entropy` - First bins the values of `x` into `max_bins` equidistant bins.
- `c3` - Uses c3 statistics to measure non linearity in the time series
- `change_quantiles` - First fixes a corridor given by the quantiles `ql` and `qh` of the distribution of `x`.
- `cid_ce` - This function calculator is an estimate for a time series complexity.
- `count_above` - Returns the percentage of values in `x` that are higher than `t`
- `count_above_mean` - Returns the number of values in `x` that are higher than the mean of `x`
- `count_below` - Returns the percentage of values in `x` that are lower than `t`
- `count_below_mean` - Returns the number of values in `x` that are lower than the mean of `x`
- `cwt_coefficients` - Calculates a Continuous wavelet transform for the Ricker wavelet, also known as the "Mexican hat wavelet" which is defined by
- `energy_ratio_by_chunks` - Calculates the sum of squares of chunk `i` out of `N` chunks expressed as a ratio with the sum of squares over the whole series.
- `fft_aggregated` - Returns the spectral centroid (mean), variance, skew, and kurtosis of the absolute fourier transform spectrum.

# TSFresh Features

- `fft_coefficient` - Calculates the fourier coefficients of the one-dimensional discrete Fourier Transform for real input by fast fourier transformation algorithm
- `first_location_of_maximum` - Returns the first location of the maximum value of x.
- `first_location_of_minimum` - Returns the first location of the minimal value of x.
- `fourier_entropy` - Calculate the binned entropy of the power spectral density of the time series (using the welch method).
- `friedrich_coefficients` - Coefficients of polynomial, which has been fitted to the deterministic dynamics of Langevin model
- `has_duplicate` - Checks if any value in x occurs more than once
- `has_duplicate_max` - Checks if the maximum value of x is observed more than once
- `has_duplicate_min` - Checks if the minimal value of x is observed more than once
- `index_mass_quantile` - Calculates the relative index i of time series x where q% of the mass of x lies left of i.
- `kurtosis` - Returns the kurtosis of x.
- `large_standard_deviation` - Does time series have large standard deviation?
- `last_location_of_maximum` - Returns the relative last location of the maximum value of x.
- `last_location_of_minimum` - Returns the last location of the minimal value of x.
- `lempel_ziv_complexity` - Calculate a complexity estimate based on the Lempel-Ziv compression algorithm.
- `length` - Returns the length of x
- `linear_trend` - Calculate a linear least-squares regression for the values of the time series versus the sequence from 0 to length of the time series minus one.
- `linear_trend_timewise` - Calculate a linear least-squares regression for the values of the time series versus the sequence from 0 to length of the time series minus one.
- `longest_strike_above_mean` - Returns the length of the longest consecutive subsequence in x that is bigger than the mean of x
- `longest_strike_below_mean` - Returns the length of the longest consecutive subsequence in x that is smaller than the mean of x

# TSFresh Features

- fast fourier matrix\_profile - Calculates the 1-D Matrix Profile[1] and returns Tukey's Five Number Set plus the mean of that Matrix Profile.
- max\_langevin\_fixed\_point - Largest fixed point of dynamics  $\text{argmax}_x \{h=0\}$  estimated from polynomial, which has been fitted to the deterministic dynamics of Langevin model
- maximum - Calculates the highest value of the time series x.
- mean - Returns the mean of x
- mean\_abs\_change - Average over first differences.
- mean\_change - Average over time series differences.
- mean\_n\_absolute\_max - Calculates the arithmetic mean of the n absolute maximum values of the time series.
- mean\_second\_derivative\_central - Returns the mean value of a central approximation of the second derivative
- median - Returns the median of x
- minimum - Calculates the lowest value of the time series x.
- number\_crossing\_m - Calculates the number of crossings of x on m.
- number\_cwt\_peaks - Number of different peaks in x.
- number\_peaks - Calculates the number of peaks of at least support n in the time series x.
- partial\_autocorrelation - Calculates the value of the partial autocorrelation function at the given lag.
- percentage\_of\_reoccurring\_datapoints\_to\_all\_datapoints - Returns the percentage of non-unique data points.
- percentage\_of\_reoccurring\_values\_to\_all\_values - Returns the percentage of values that are present in the time series more than once.
- permutation\_entropy - Calculate the permutation entropy.
- quantile - Calculates the q quantile of x.
- query\_similarity\_count - This feature calculator accepts an input query subsequence parameter, compares the query (under z-normalized Euclidean distance) to all subsequences within the time series, and returns a count of the number of times the query was found in the time series (within some predefined maximum distance threshold).

# TSFresh Features

- `range_count` - Count observed values within the interval [min, max).
- `ratio_beyond_r_sigma` - Ratio of values that are more than  $r \cdot \text{std}$  ( $5$  or  $r$  times sigma) away from the mean of  $x$ .
- `ratio_value_number_to_time_series_length` - Returns a factor which is 1 if all values in the time series occur only once, and below one if this is not the case.
- `root_mean_square` - Returns the root mean square (rms) of the time series.
- `sample_entropy` - Calculate and return sample entropy of  $x$ .
- `set_property` - This method returns a decorator that sets the property key of the function to value
- `skewness` - Returns the sample skewness of  $x$  (calculated with the adjusted Fisher-Pearson standardized moment coefficient G1).
- `spkt_welch_density` - This feature calculator estimates the cross power spectral density of the time series  $x$  at different frequencies.
- `standard_deviation` - Returns the standard deviation of  $x$
- `sum_of_reoccurring_data_points` - Returns the sum of all data points, that are present in the time series more than once.
- `sum_of_reoccurring_values` - Returns the sum of all values, that are present in the time series more than once.
- `sum_values` - Calculates the sum over the time series values
- `symmetry_looking` - Boolean variable denoting if the distribution of  $x$  looks symmetric.
- `time_reversal_asymmetry_statistic` - Returns the time reversal asymmetry statistic.
- `value_count` - Count occurrences of value in time series  $x$ .
- `variance` - Returns the variance of  $x$
- `variance_larger_than_standard_deviation` - Is variance higher than the standard deviation?
- `variation_coefficient` - Returns the variation coefficient (standard error / mean, give relative value of variation around mean) of  $x$

# catch22: CAnonical Time-series CHaracteristics

---

- The catch22 feature set spans a diverse range of time-series characteristics representative of the diversity of interdisciplinary methods for TSA.
- Features in catch22 capture TS properties of the distribution of values in the TS, linear and nonlinear temporal autocorrelation properties, scaling of fluctuations, and others.
- Selected by applying the procedure describe in [Lubba 2019] to a set of 93 datasets containing over 147k TS and using a filtered version of the HCTSA feature library (4791 features).
- The reduction from 4791 to 22 features is associated with a 1000-fold reduction in computation time and near linear scaling with TS length, despite an average reduction in classification accuracy of just 7%.

### Distribution

|                     |                                                     |
|---------------------|-----------------------------------------------------|
| DN_HistogramMode_5  | Mode of $z$ -scored distribution (5-bin histogram)  |
| DN_HistogramMode_10 | Mode of $z$ -scored distribution (10-bin histogram) |

### Simple temporal statistics

|                                  |                                                                 |
|----------------------------------|-----------------------------------------------------------------|
| SB_BinaryStats_mean_longstretch1 | Longest period of consecutive values above the mean             |
| DN_OutlierInclude_p_001_mdrmd    | Time intervals between successive extreme events above the mean |
| DN_OutlierInclude_n_001_mdrmd    | Time intervals between successive extreme events below the mean |

### Linear autocorrelation

|                                  |                                                                          |
|----------------------------------|--------------------------------------------------------------------------|
| CO_f1ecac                        | First $1/e$ crossing of autocorrelation function                         |
| CO_FirstMin_ac                   | First minimum of autocorrelation function                                |
| SP_Summaries_welch_rect_area_5_1 | Total power in lowest fifth of frequencies in the Fourier power spectrum |
| SP_Summaries_welch_rect_centroid | Centroid of the Fourier power spectrum                                   |
| FC_LocalSimple_mean3_stderr      | Mean error from a rolling 3-sample mean forecasting                      |

### Nonlinear autocorrelation

|                                         |                                                                     |
|-----------------------------------------|---------------------------------------------------------------------|
| CO_trev_1_num                           | Time-reversibility statistic, $\langle (x_{t+1} - x_t)^3 \rangle_t$ |
| CO_HistogramAMI_even_2_5                | Automutual information, $m = 2, \tau = 5$                           |
| IN_AutoMutualInfoStats_40_gaussian_fmml | First minimum of the automutual information function                |

### Successive differences

|                                      |                                                                                  |
|--------------------------------------|----------------------------------------------------------------------------------|
| MD_hrv_classic_pnn40                 | Proportion of successive differences exceeding $0.04\sigma$ [20]                 |
| SB_BinaryStats_diff_longstretch0     | Longest period of successive incremental decreases                               |
| SB_MotifThree_quantile_hh            | Shannon entropy of two successive letters in equiprobable 3-letter symbolization |
| FC_LocalSimple_mean1_tauresrat       | Change in correlation length after iterative differencing                        |
| CO_Embed2_Dist_tau_d_expfit_meandiff | Exponential fit to successive distances in 2-d embedding space                   |

### Fluctuation Analysis

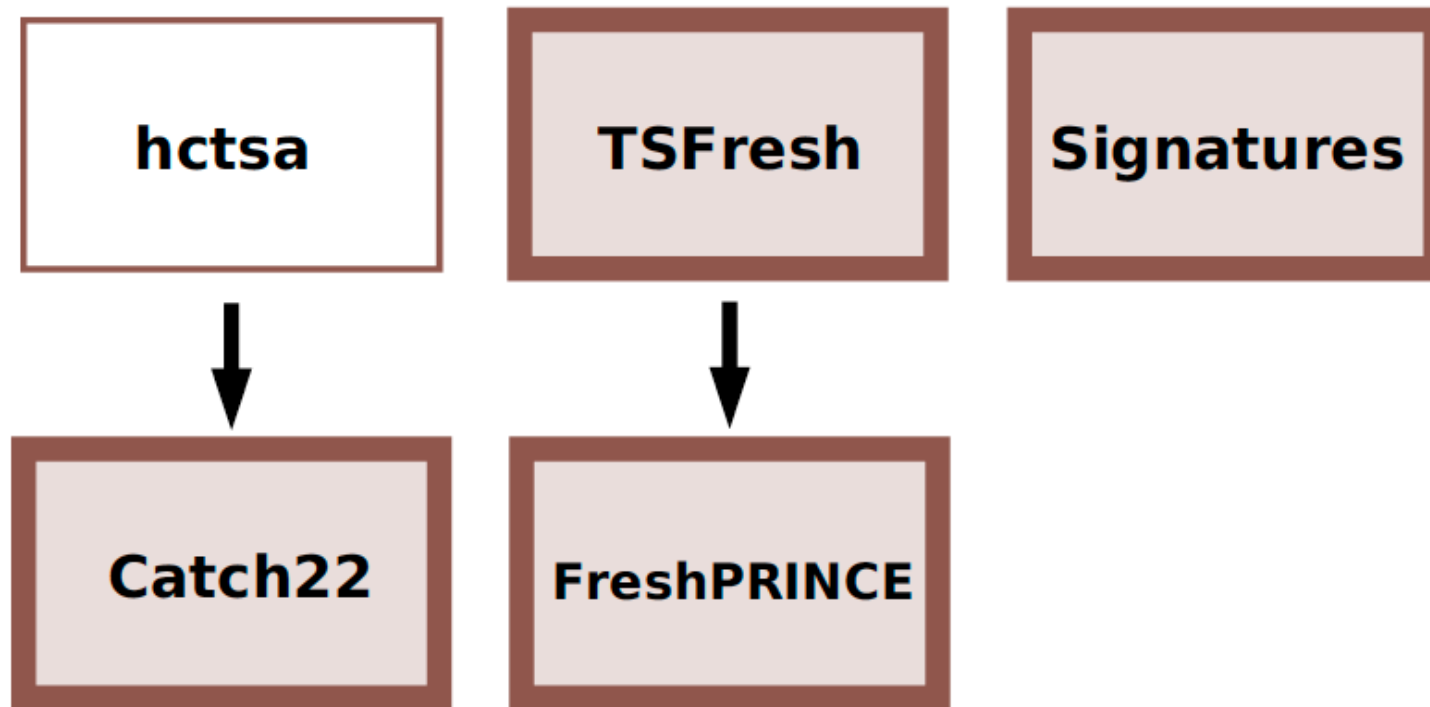
|                                             |                                                                                          |
|---------------------------------------------|------------------------------------------------------------------------------------------|
| SC_FluctAnal_2_dfa_50_1_2_logi_prop_r1      | Proportion of slower timescale fluctuations that scale with DFA (50% sampling)           |
| SC_FluctAnal_2_rsrangefit_50_1_logi_prop_r1 | Proportion of slower timescale fluctuations that scale with linearly rescaled range fits |

### Others

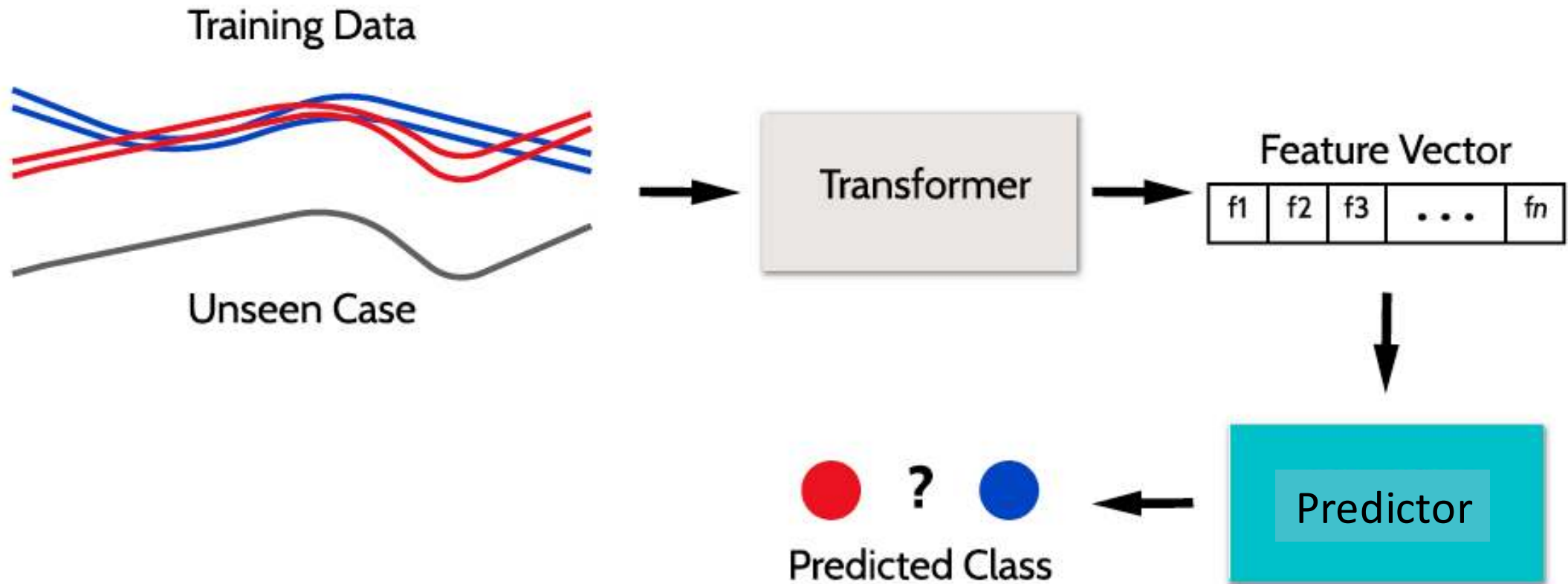
|                                    |                                                                               |
|------------------------------------|-------------------------------------------------------------------------------|
| SB_TransitionMatrix_3ac_sumdiagcov | Trace of covariance of transition matrix between symbols in 3-letter alphabet |
| PD_PeriodicityWang_th0_01          | Periodicity measure of [31]                                                   |

# Overview of Global Features and Relationships

---



# Global Feature-based Predictor



# Features, Approximations, Distances and Normalizations

---

- Normalizations can be applied before global features extraction depending on the objective of your TSA task.
- Time-Dependent approximations can be applied before global features extraction depending on the objective of your TSA task.
- It does not make any sense to use Time-Independent approximation after that global features have been extracted.
- It does not make any sense to use a distance function accounting for time like DTW after that global features have been extracted.

# References

---

- Time Series Analysis and Its Applications. Robert H. Shumway and David S. Stoffer. 4<sup>th</sup> edition. (<http://www.stat.ucla.edu/~frederic/415/S23/tsa4.pdf>)
- Mining Time Series Data. Chotirat Ann Ratanamahatana et al. 2010. ([https://www.researchgate.net/publication/227001229\\_Mining\\_Time\\_Series\\_Data](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
- catch22: CANonical Time-series CHaracteristics selected through highly comparative time-series analysis Carl H Lubba et al. 2019.