Time Series Management

Michele Linardi Ph.D.

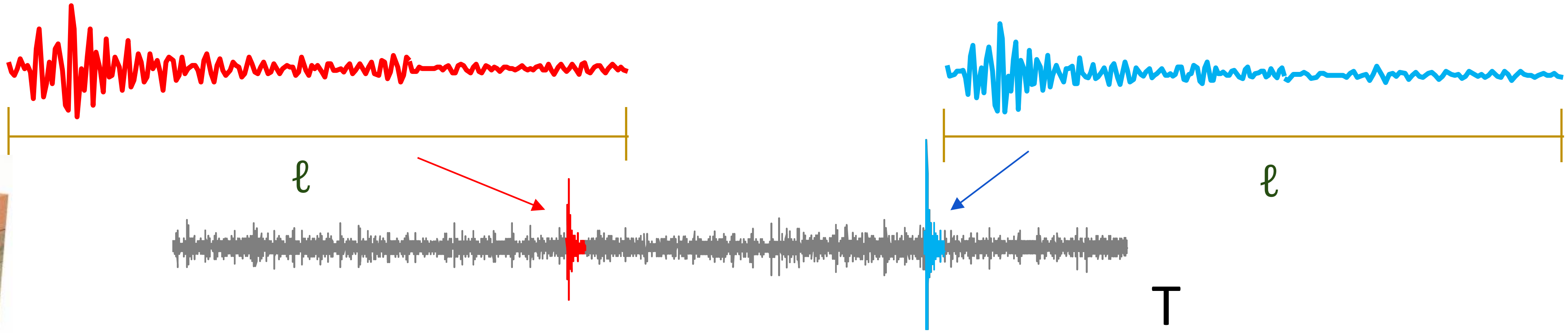
michele.linardi@orange.fr

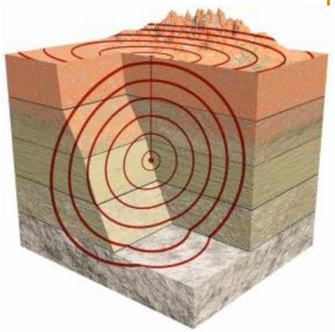
Syllabus

- Time series data mining
- Motif primitive
- Discord (Outlier)
- Matrix profile Algorithm
- Algorithm Python code DEMO

Data Series Motifs

- **Motif discovery** is one of the most useful primitives for data series mining
- Given a time series T, a motif is the pair of subsequences of length ℓ with the smallest Euclidean distance.

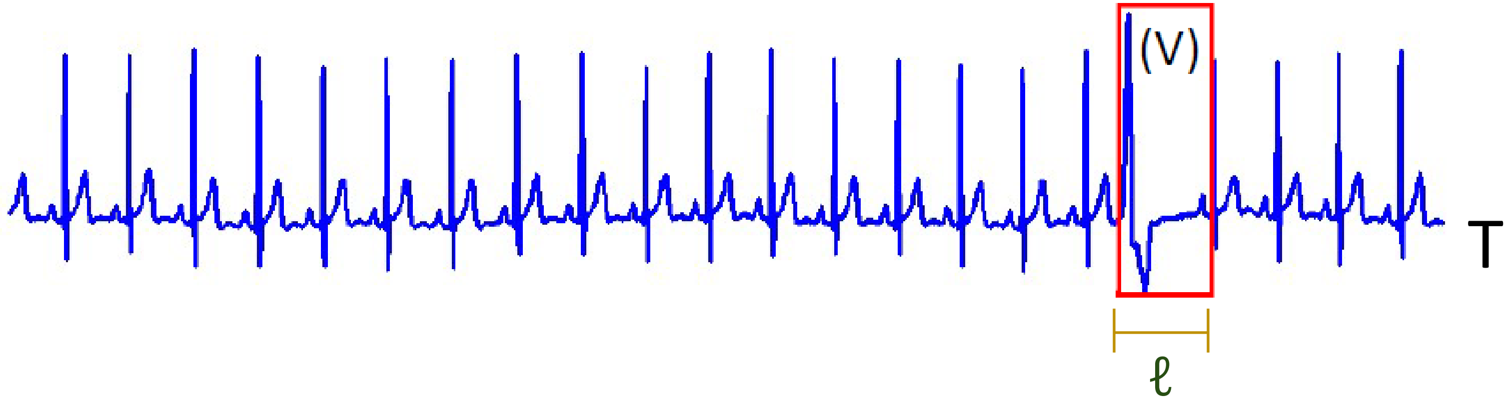




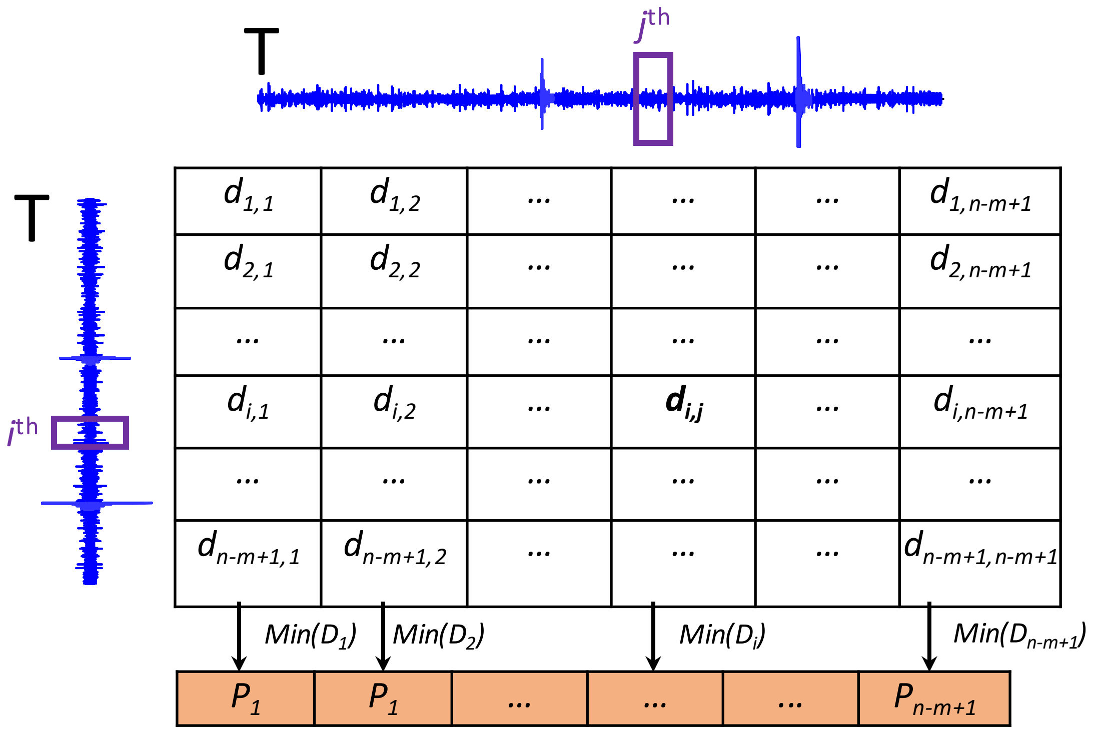
Data Series Discords

- **Discords** are used to detect anomalous/interesting patterns
  - usually represent the most unusual subsequences within a **time series**

- Given a time series T, a discord is the subsequence of length ℓ that has the largest Euclidean distance to its nearest neighbor.



Matrix Profile - Motif and Discord Discovery

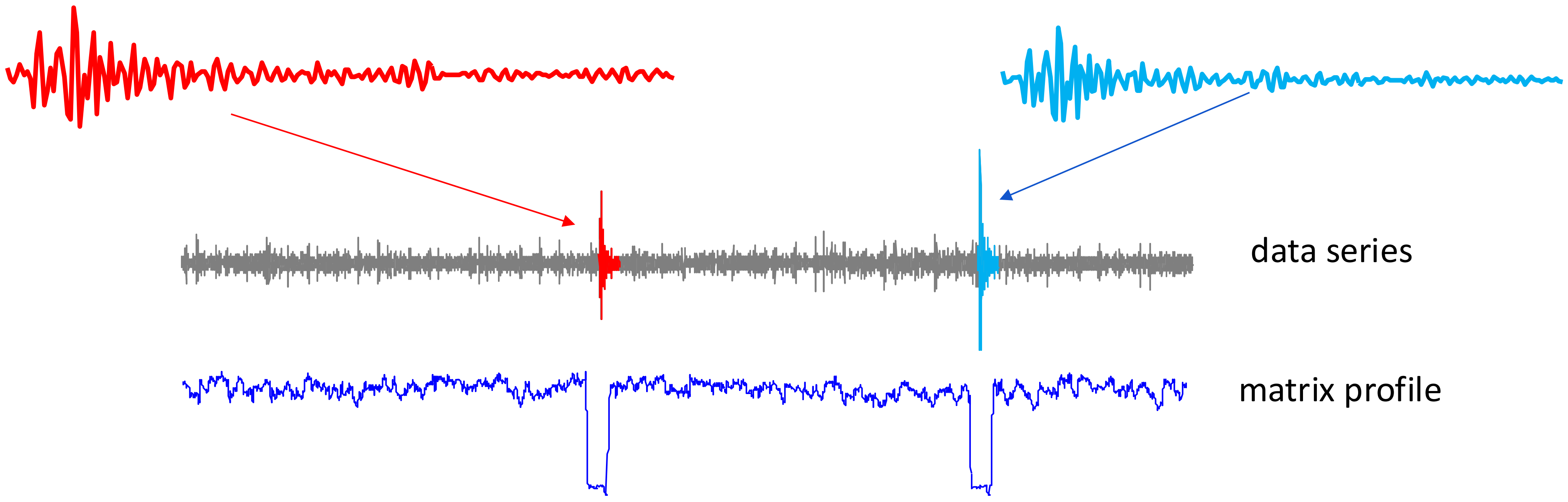


Distance Matrix: a symmetric matrix, which contains the pairwise subsequences (of length ℓ) distances in T.

Excluding Trivial matches : = distances of any sequence pair having indexes (e.g., i and j) closer than a given threshold

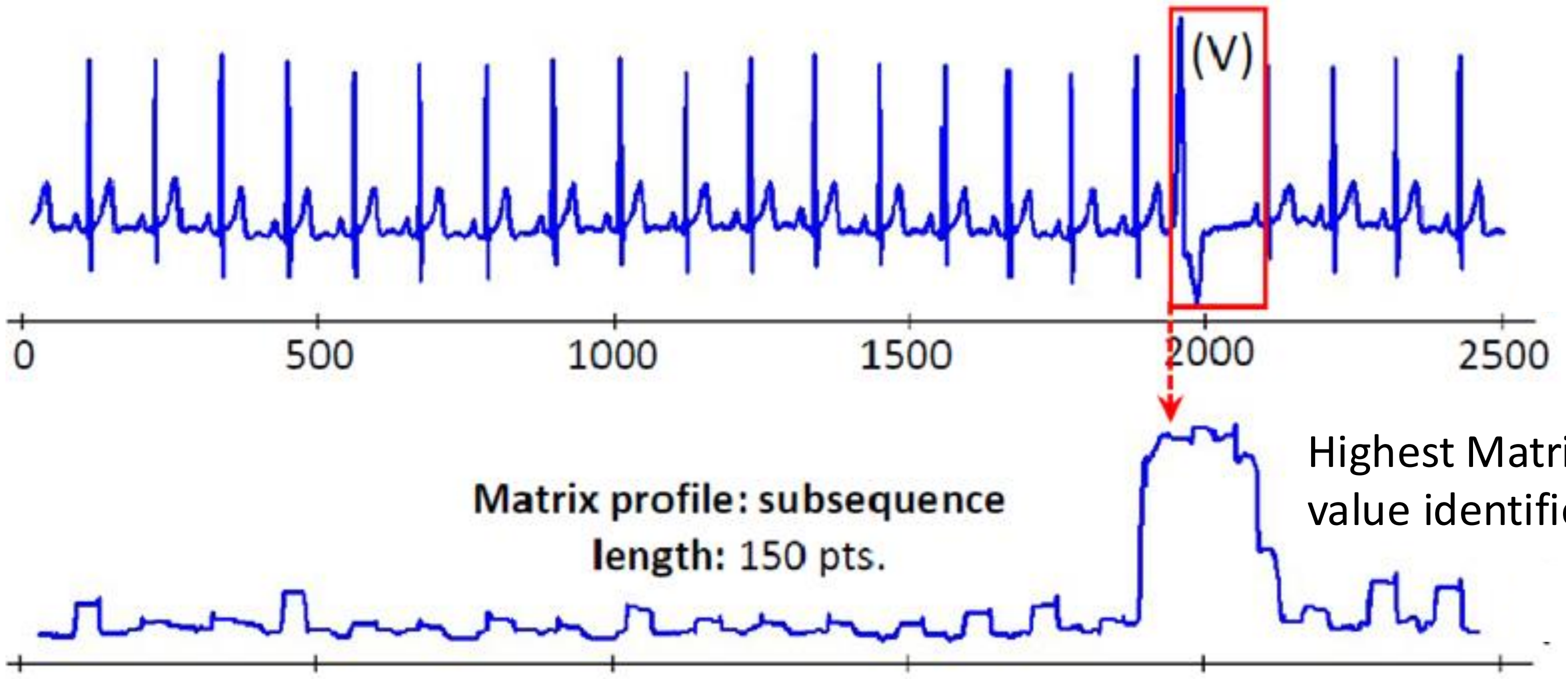
Matrix Profile: a vector of distance between each subsequence and its nearest neighbor

Matrix Profile - Motif and Discord Discovery



A pair of minimum points identifies the Motif

Matrix Profile - Motif and Discord Discovery



Highest Matrix Profile
value identifies Discord

Euclidean distance metrics

Given two time series

x = $x_1 \ldots x_n$

and

y = $y_1 \ldots y_n$

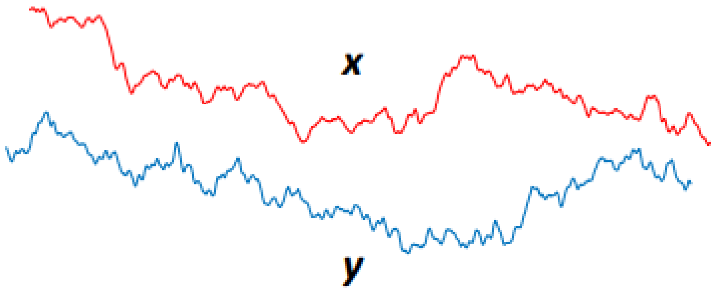
their z-Normalized Euclidean distance is defined as:

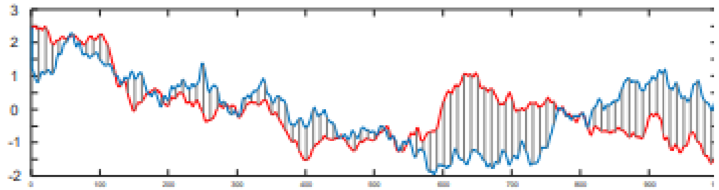
$$\hat{x}_i = \frac{x_i - \mu_x}{\sigma_x} \qquad \hat{y}_i = \frac{y_i - \mu_y}{\sigma_y}$$

```
function y = zNorm(x)
y = (x-mean(x))/std(x,1);
```

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

```
function d = EuclideanDistance(x,y)
d = sqrt(sum((x-y).^2));
```





Pearson correlation coefficient

- Given two time series $\boldsymbol{x}$ and $\boldsymbol{y}$ of length m.
- Correlation Coefficient:

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

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

- Sufficient Statistics:

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

The sufficient statistics can be calculated in one linear scan. Given the sufficient statistics, correlation coefficient is a constant operation. Note the use of the dot product, which is the key component of many lockstep measures.

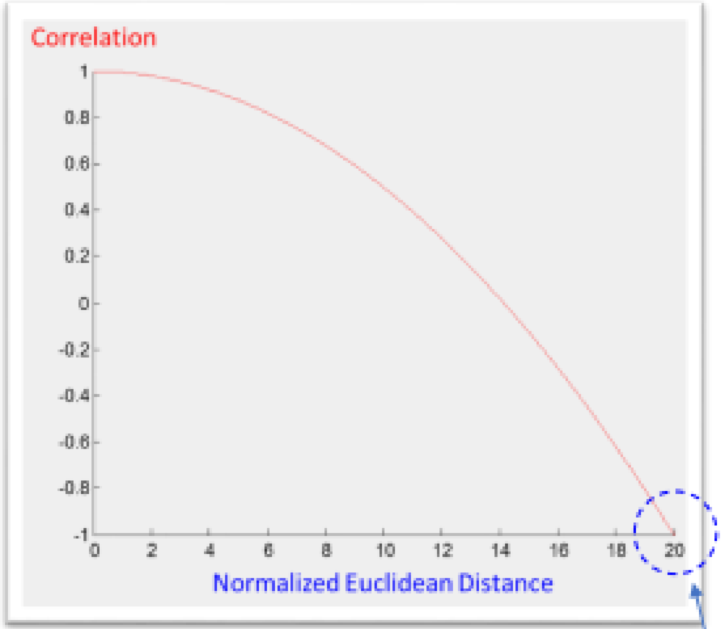
Correlation – Euclidean Distance relationship

Z-normalized Euclideand distance

$\boldsymbol{m}$:= number of data points

$$d(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) = \sqrt{2m(1 - corr(\boldsymbol{x}, \boldsymbol{y}))}$$

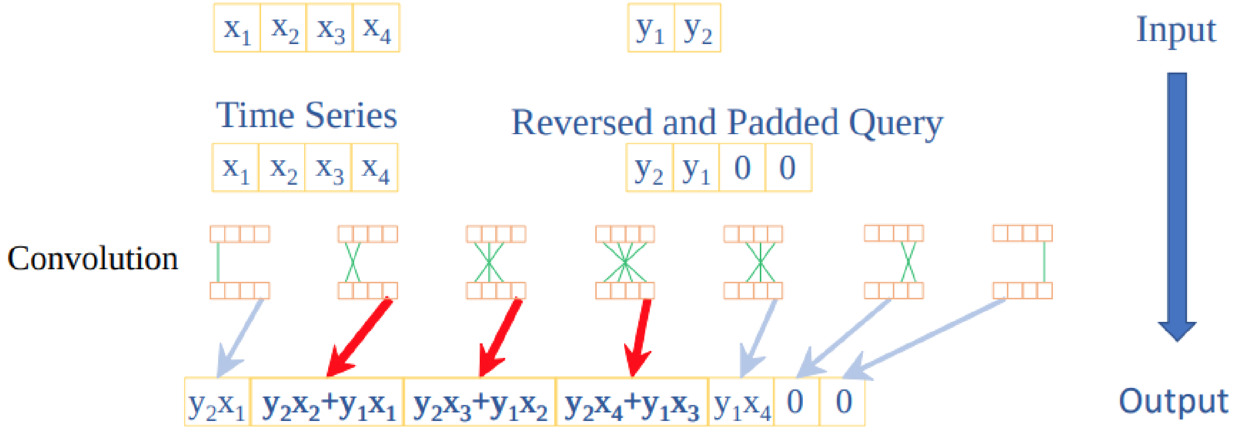
- Correlation coefficient does not obey triangular inequality, while Euclidean distance does
- Maximizing correlation coefficient can be achieved by minimizing normalized Euclidean distance and vice versa
- Correlation coefficient is bounded between -1 and 1, while z-normalized Euclidean distance is bounded between zero and a positive number dependent on m



20 for m = 100

Mueen’s Algorithm for Similarity Search (MASS)

- MASS uses a convolution based method to calculate sliding dot products in $O(n \log n)$
- Convolution: If x and y are vectors of polynomial coefficients, convolving them is equivalent to multiplying the two polynomials.
- We can use convolution to compute all of the sliding dot products between the query and sliding windows.
- Convolution can be computed in the frequency domain, using the **Fast Fourier Transform**.



Matrix profile algorithm – STOMP (1/6)

Scalable Time series Ordered Matrix Profile

O(n^2) time, O(n) space algorithm to compute the matrix profile.

Matrix profile algorithm – STOMP (2/6)

Scalable Time series Ordered Matrix Profile

$$T_iT_j = \sum_{k=0}^{m-1} t_{i+k}t_{j+k}$$ Dot product of the i^{th} window and the j^{th} window.

Z-normalized Euclideand distance

$\boldsymbol{m}$:= number of data points

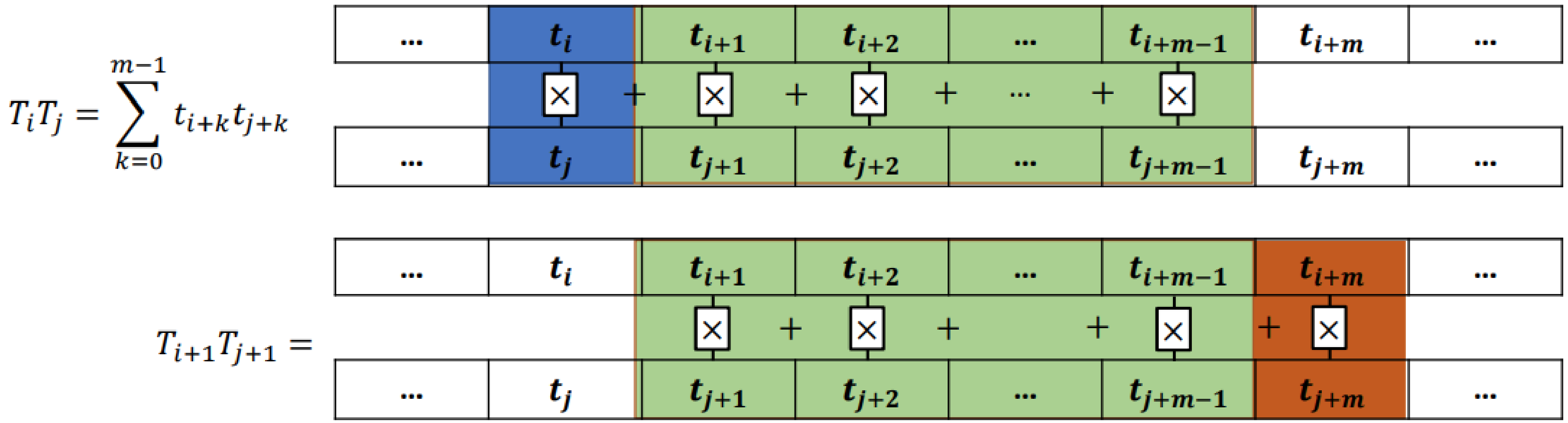
$$d_{i,j} = \sqrt{2m\left(1 - \frac{T_iT_j - m\mu_i\mu_j}{m\sigma_i\sigma_j}\right)}$$

- We can precompute and store the means and standard deviations in O(n) space and time
- Once we know T_i, T_j , it takes O(1) time to compute the distance $d_{i,j}$

Matrix profile algorithm – STOMP (3/6)

Scalable Time series Ordered Matrix Profile

- The relationship between $T_i T_j$ and $T_{i+1} T_{j+1}$



$$T_{i+1}T_{j+1} = T_i T_j - t_i t_j + t_{i+m} t_{j+m}$$

$O(1)$ time complexity

Matrix profile algorithm – STOMP (4/6)

Scalable Time series Ordered Matrix Profile

1) All means and standard deviations are precomputed. This costs linear time and space.

μ_1	μ_2	μ_3	$\dots$	μ_{n-m+1}

σ_1	σ_2	σ_3	$\dots$	σ_{n-m+1}

Matrix profile algorithm – STOMP (5/6)

Scalable Time series Ordered Matrix Profile

2) The first column and row of the matrix are identical and pre-computed by MASS.

μ_1	μ_2	μ_3	...	μ_{n-m+1}

σ_1	σ_2	σ_3	...	σ_{n-m+1}

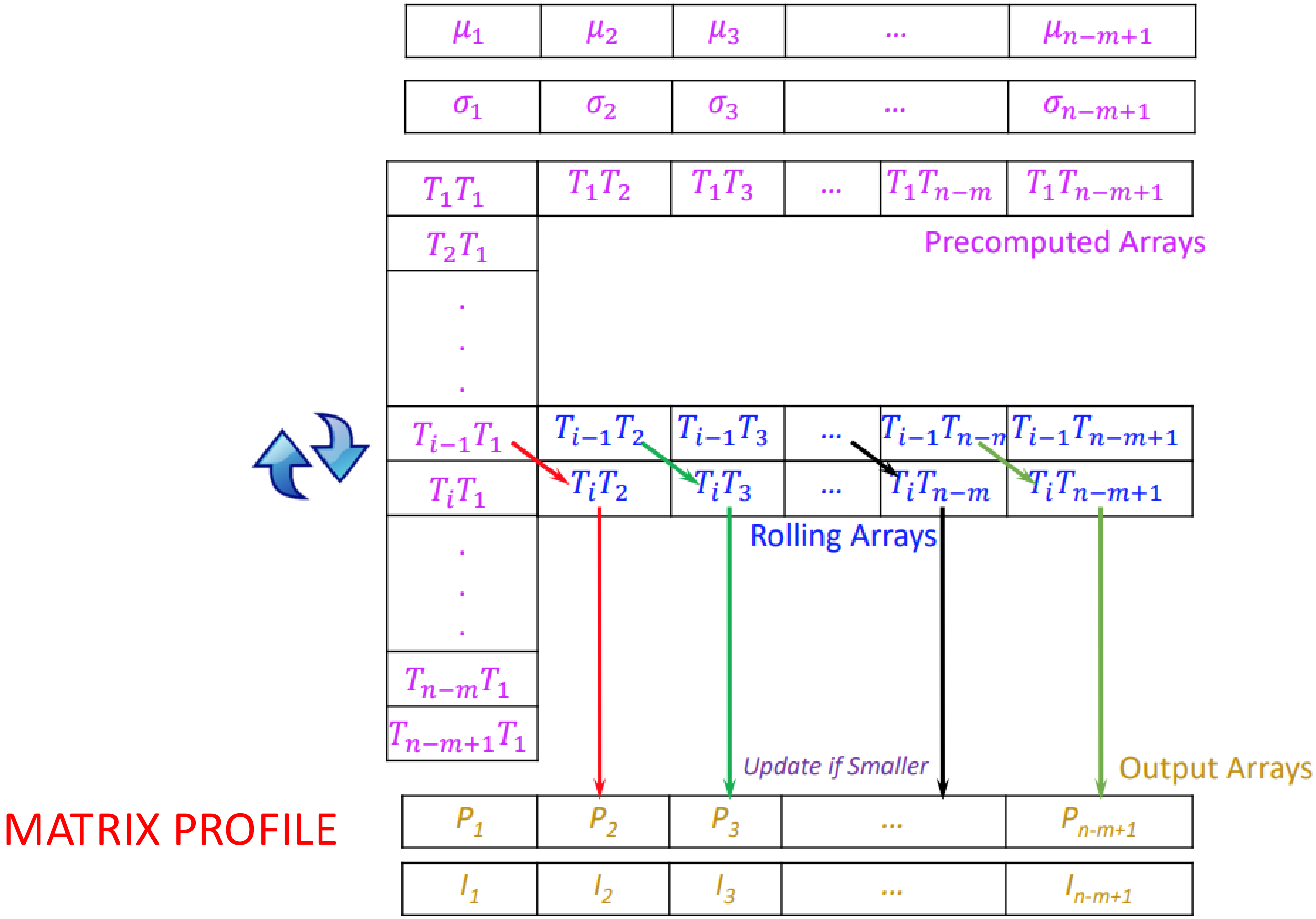
T_1T_1	T_1T_2	T_1T_3	...	T_1T_{n-m}	T_1T_{n-m+1}
T_2T_1					

Precomputed Arrays

Matrix profile algorithm – STOMP (6/6)

Scalable Time series Ordered Matrix Profile

3) The algorithm iterates through the rows. The previous row is maintained as a local array to feed dot products.



Notebook motif discovery

- https://github.com/target/matrixprofile-ts/blob/master/docs/examples/Motif%20Discovery.ipynb

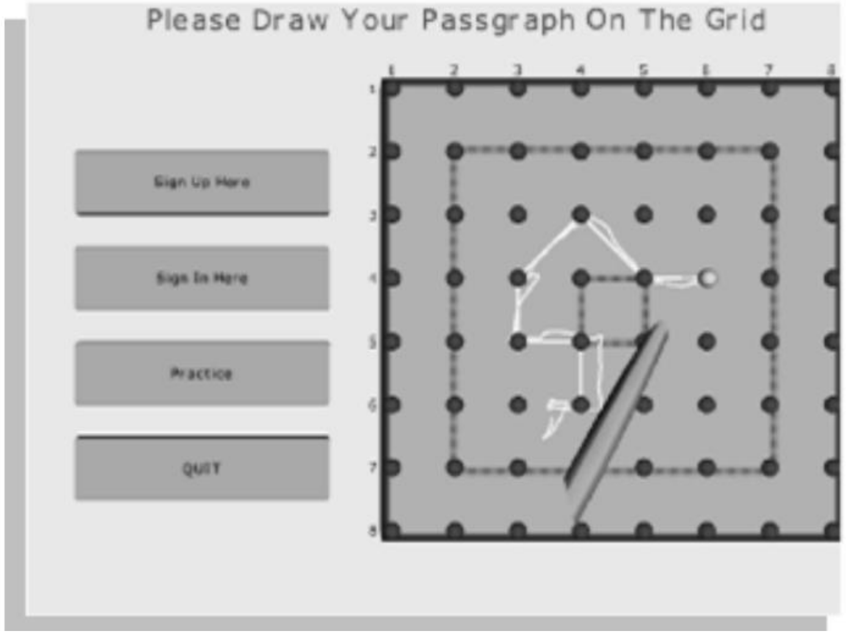
- Matrix Profile library:

https://github.com/target/matrixprofile-ts/tree/master

Presentation du TD

Haptics DATA

Data are taken from 5 people entering their passgraph (a code to access a system protected by a graphical authentication system) on a touchscreen. The data are the x-axis movement only.



Assignment

- Open the Python Notebook

TS_MotifDiscovery.ipynb

Load the modules in the folder

- Instruction are contained in the notebook

References

- Time Series Data Mining Using the Matrix Profile: A Unifying View of Motif Discovery, Anomaly Detection, Segmentation, Classification, Clustering and Similarity Joins www.cs.ucr.edu/~eamonn/MatrixProfile.html
- https://stumpy.readthedocs.io/en/latest/Tutorial_The_Matrix_Profile.html
- Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets https://ieeexplore.ieee.org/document/7837992
- https://github.com/matrix-profile-foundation