Time Series - Shapelet/Motif Discovery

UNIVERSITÀ DI PISA

Shapelet

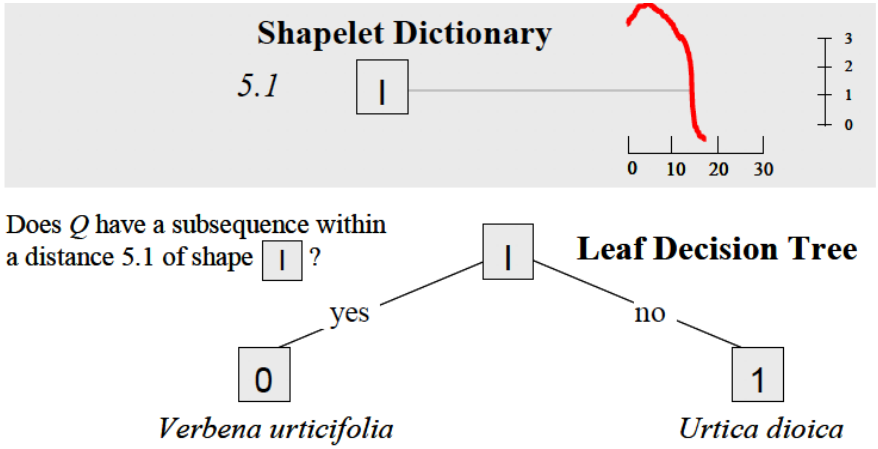
Time Series Classification

- Given a set X of n time series, $X = \{x_1, x_2, \ldots, x_n\}$, each time series has m ordered values $x_i = < x_{t1}, x_{t2}, \ldots, x_{tm} >$ and a class value c_i.
- The objective is to find a function f that maps from the space of possible time series to the space of possible class values.
- Generally, it is assumed that all the TS have the same length m.

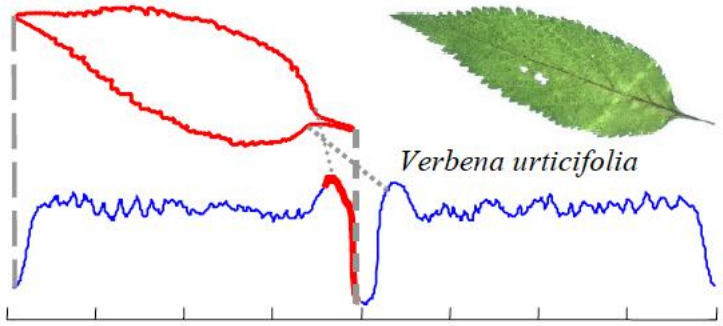
Shapelet-based Classification

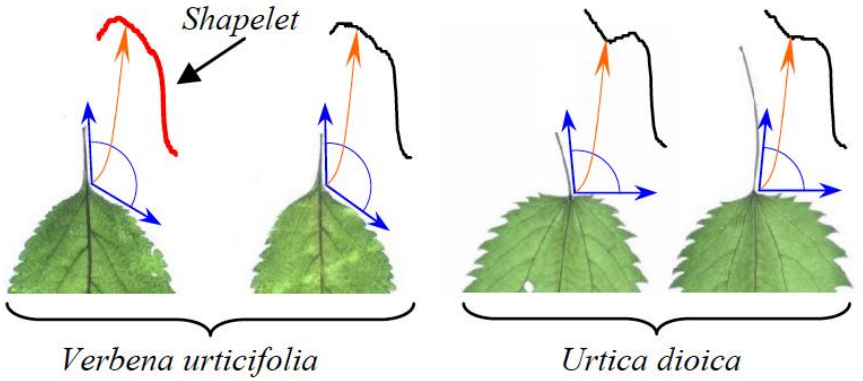
1. Represent a TS as a vector of distances with representative subsequences, namely shapelets.

2. Use it as input for machine learning classifiers.



3.2	8.7
1.4	7.9
6.7	4.2
9.2	3.4

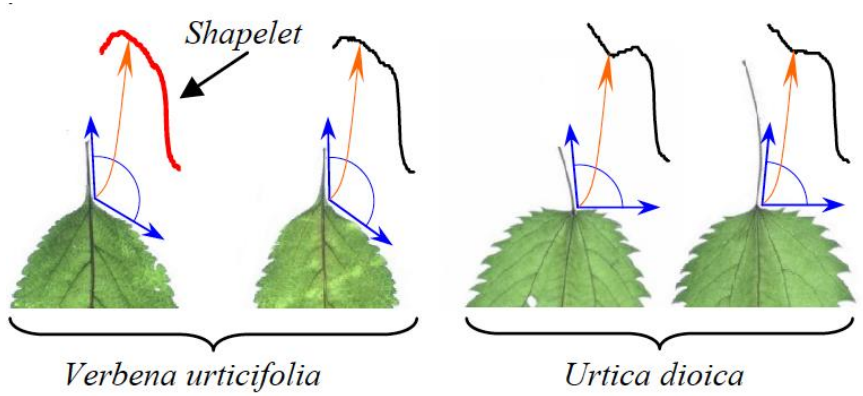




Time Series Shapelets

- Shapelets are TS subsequences which are maximally representative of a class.
- Shapelets can provide interpretable results, which may help domain practitioners better understand their data.
- Shapelets can be significantly more accurate/robust because they are *local features*, whereas most other state-of-the-art TS classifiers consider *global features*.

Verbena 0.87
Urtica 0.34



Finding Shapelets

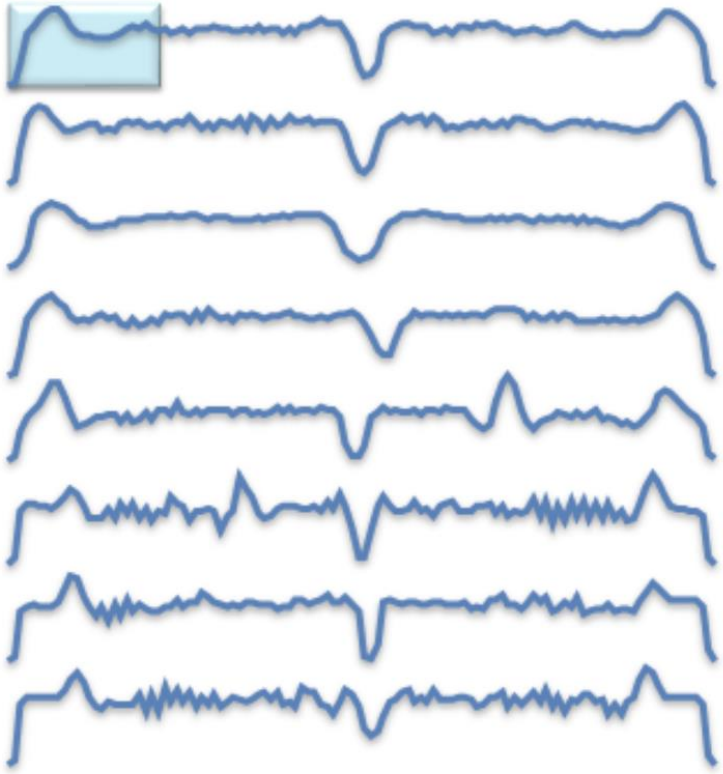
```
FindingShapeletBF (dataset D, MAXLEN, MINLEN)
1   candidates ← GenerateCandidates(D, MAXLEN, MINLEN)
2   bsf_gain ← 0
3   For each S in candidates
4       gain ← CheckCandidate(D, S)
5       If gain > bsf_gain
6           bsf_gain ← gain
7           bsf_shapelet ← S
8       EndIf
9   EndFor
10  Return bsf_shapelet
```

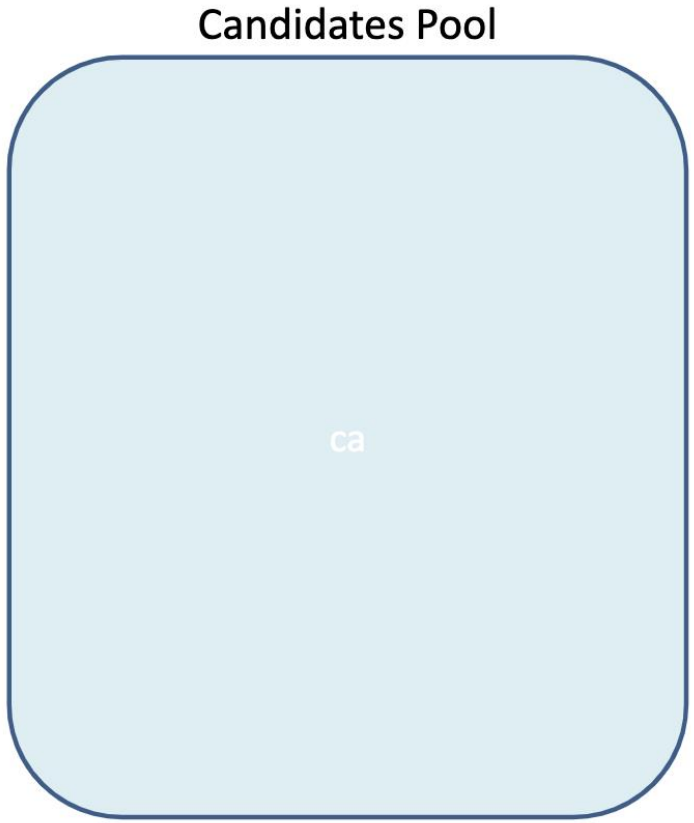
Generate Candidate

Sliding a **window of size *l*** across all of the time series objects in the dataset D, extracts all of the possible candidates and adds them to the pool

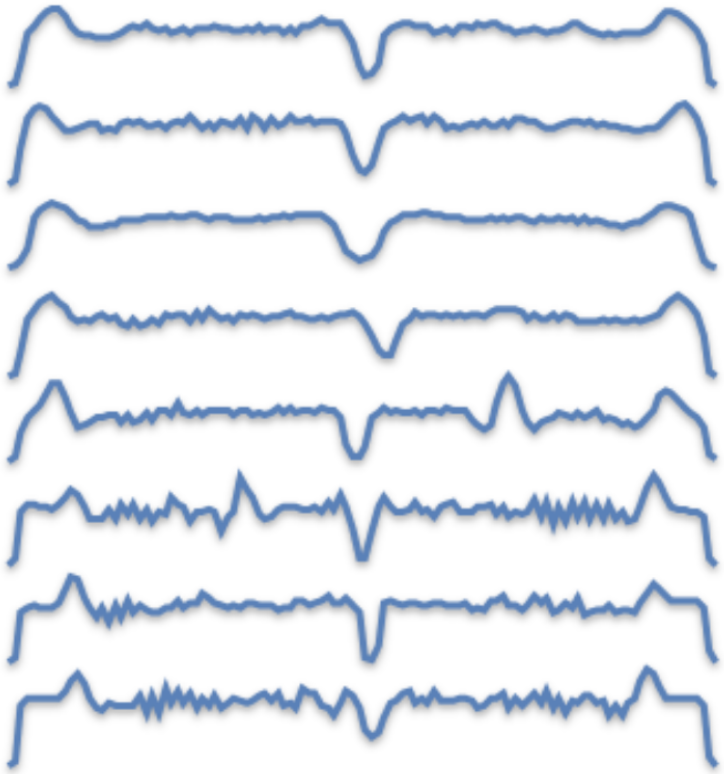
GenerateCandidates (dataset **D**, *MAXLEN*, *MINLEN*)	
1	*pool* ← Ø
2	*l* ← *MAXLEN*
3	**While** *l* ≥ *MINLEN*
4	**For** *T* **in D**
5	*pool* ← *pool* ∪ $\mathbf{S}_T^l$
6	**EndFor**
7	*l* ← *l* - 1
8	**EndWhile**
9	**Return** *pool*

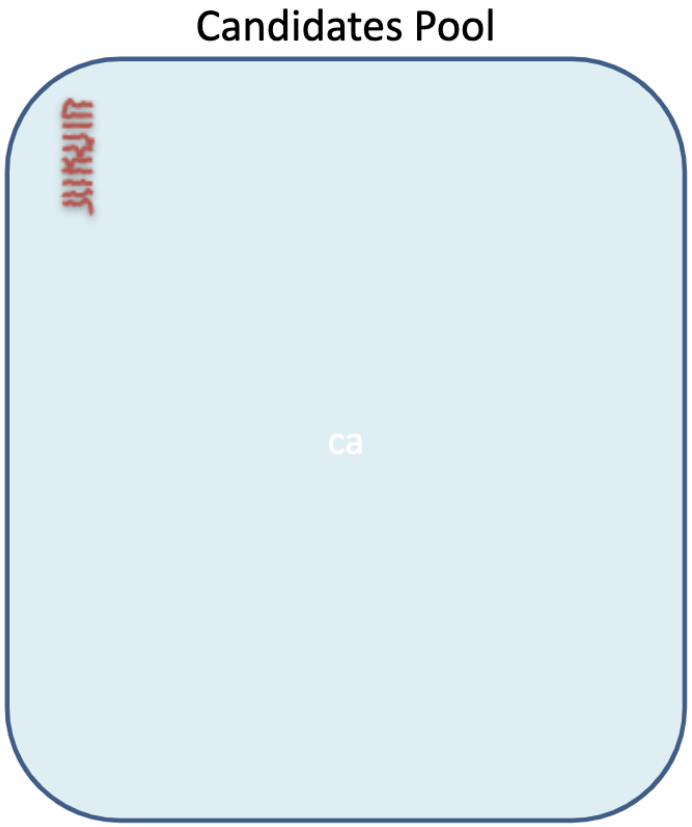
Extract Subsequences of all Possible Lengths



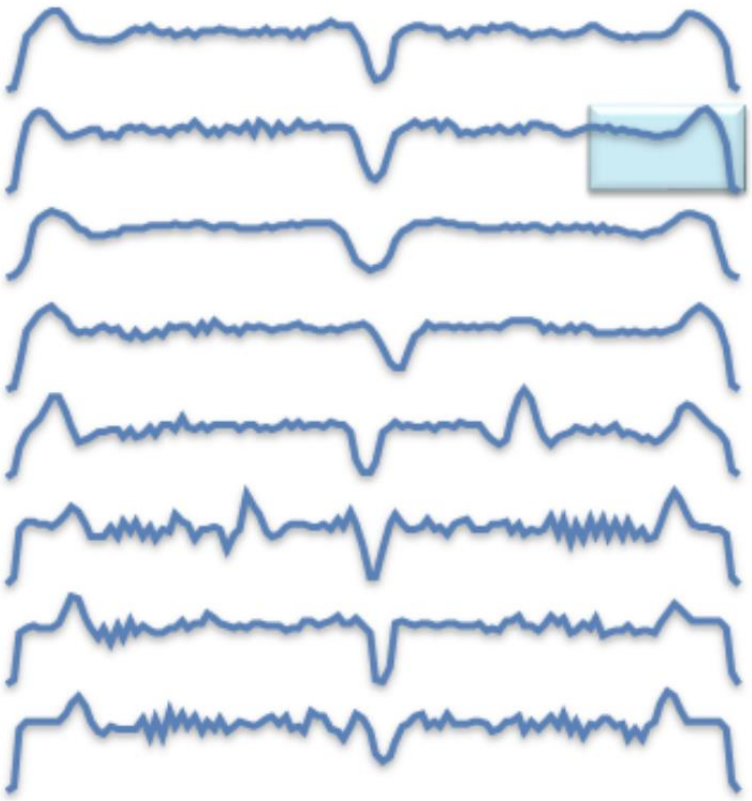


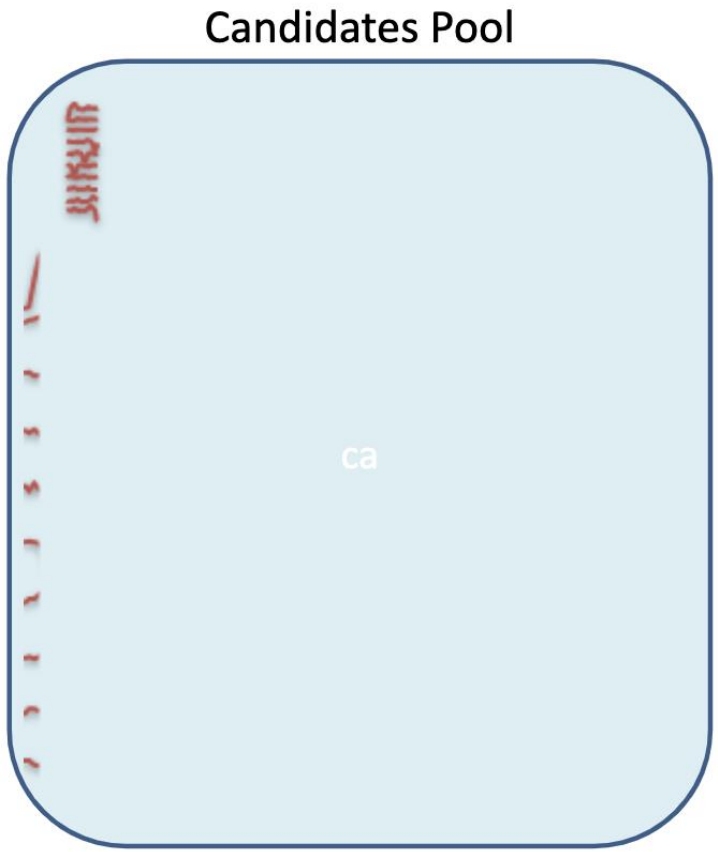
Extract Subsequences of all Possible Lengths



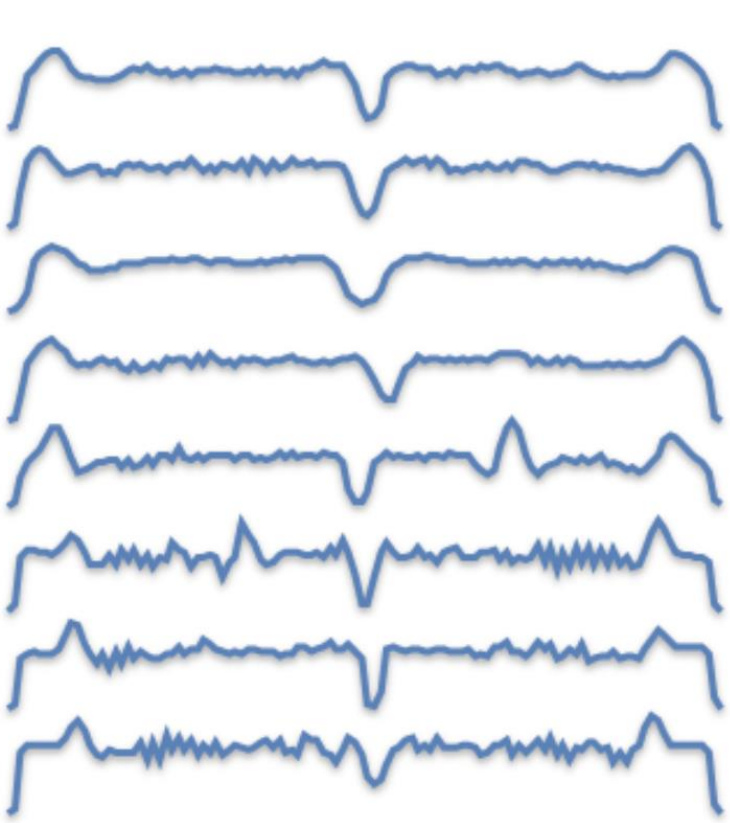


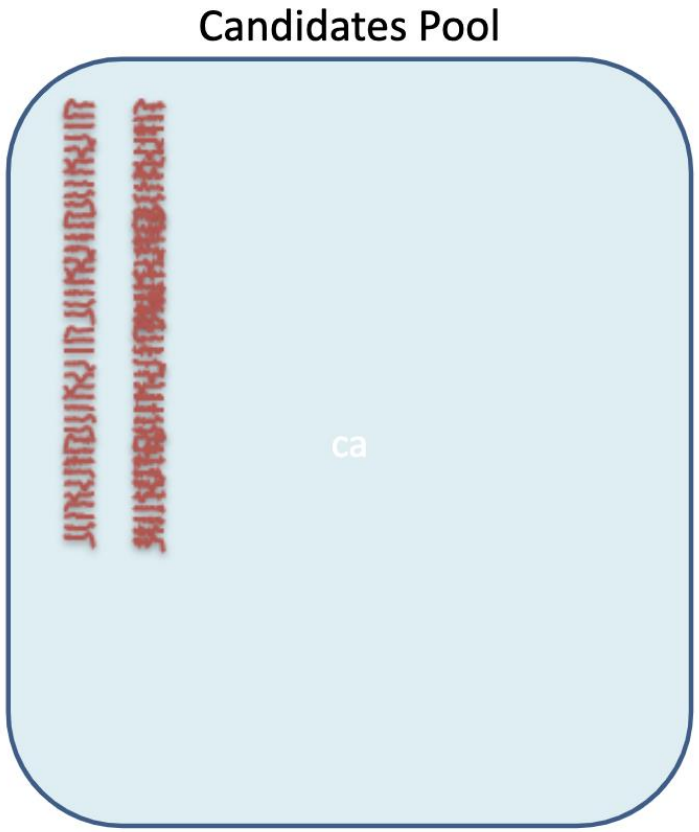
Extract Subsequences of all Possible Lengths



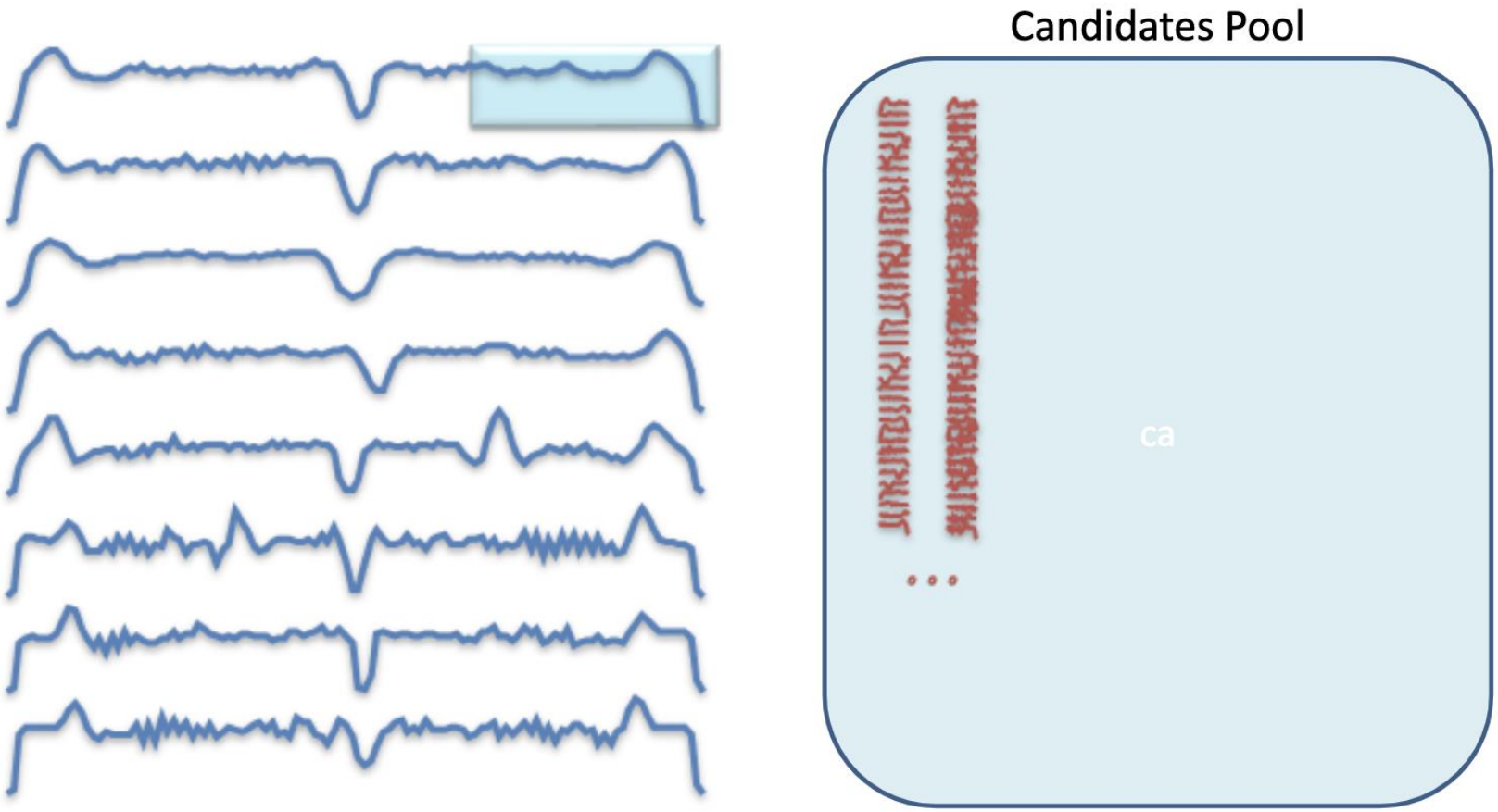


Extract Subsequences of all Possible Lengths





Extract Subsequences of all Possible Lengths



Check Candidates

CheckCandidate (dataset **D**, shapelet candidate *S*)	
1	*objects_histogram* ← Ø
2	**For each** *T* **in D**
3	*dist* ← SubsequenceDist(*T*, *S*)
4	insert *T* into *objects_histogram* by the key *dist*
5	**EndFor**
6	**Return** CalculateInformationGain(*objects_histogram*)

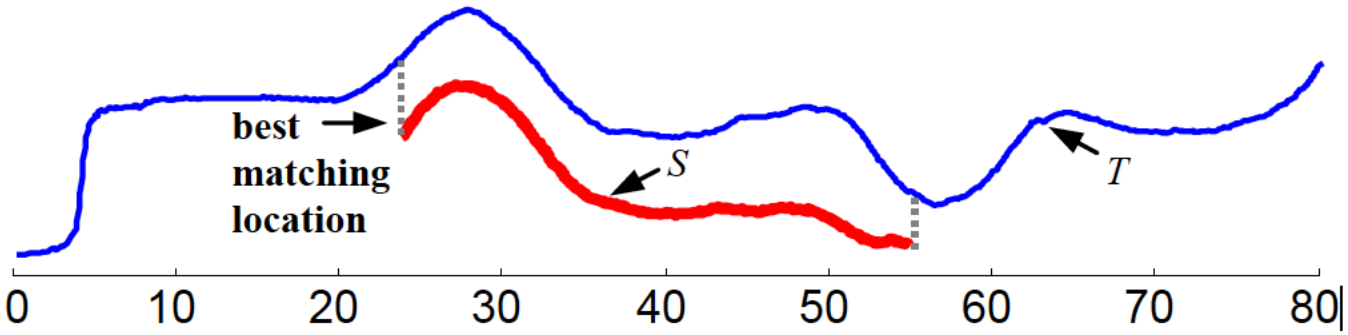
- Inserts all of the time series objects into the histogram objects_histogram according to the distance from the time series object to the candidate
- Calculate Information Gain

Distance with a Subsequence

- Distance from the TS to the subsequence *SubsequenceDist(T, S)* is a distance function that takes time series *T* and subsequence *S* as inputs and returns a non-negative value *d*, which is the distance from *T* to *S*.

- *SubsequenceDist(T, S) = min(Dist(S, S')), for* $S' \in S_T^{|S|}$
  - where $S_T^{|S|}$ is the set of all possible subsequences of *T*

- Intuitively, it is the distance between *S* and its best matching location in *T*.

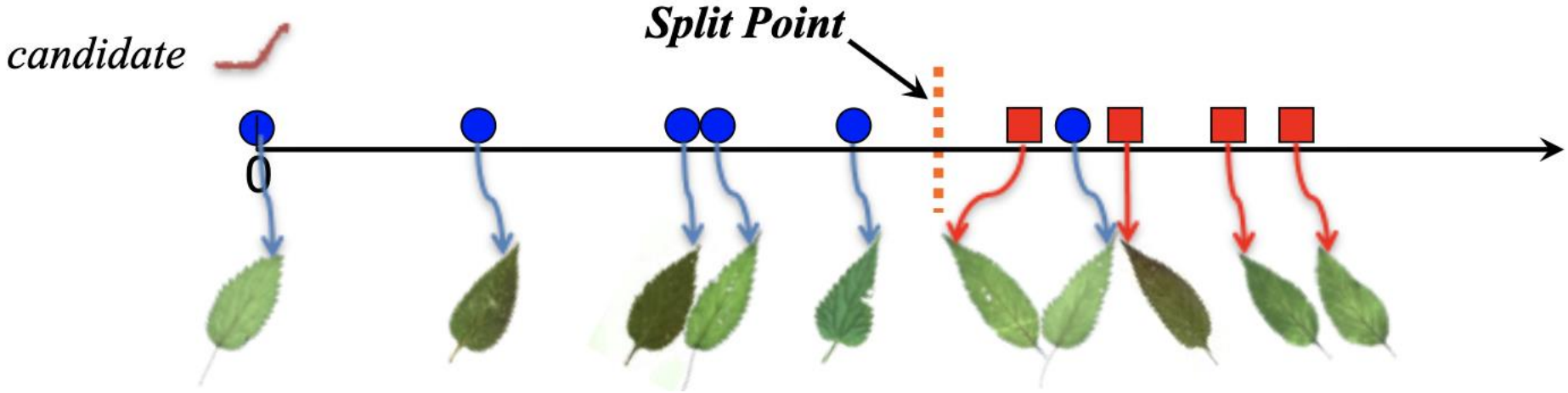


Check Candidates with IG

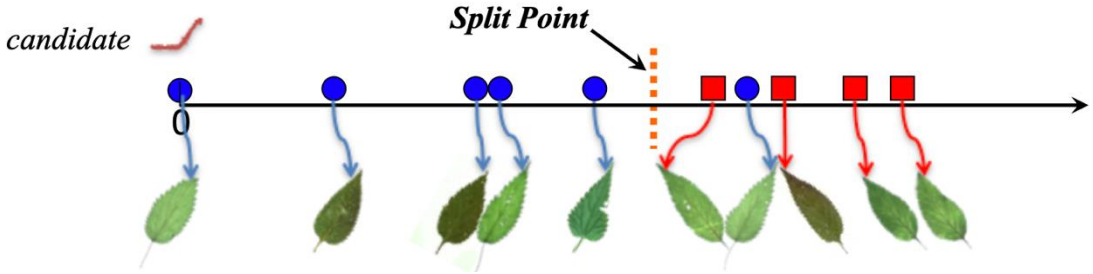
CalculateInformationGain (distance histogram *obj_hist*)	
1	*split_dist* ← OptimalSplitPoint(*obj_hist*)
2	$\mathbf{D_1} \leftarrow \emptyset$, $\mathbf{D_2} \leftarrow \emptyset$
3	**For** *d* **in** *obj_hist*
4	**If** *d.dist* < *split_dist*
5	$\mathbf{D_1} \leftarrow \mathbf{D_1} \cup d.objects$
6	**Else**
7	$\mathbf{D_2} \leftarrow \mathbf{D_2} \cup d.objects$
8	**EndIf**
9	**EndFor**
10	**Return** $I(\mathbf{D}) - \hat{I}(\mathbf{D})$

Testing The Utility of a Candidate Shapelet

- Arrange the TSs in the dataset *D* based on the distance from the candidate.
- Find the optimal split point that maximizes the information gain (same as for Decision Tree classifiers)
- Pick the candidate achieving best utility as the shapelet

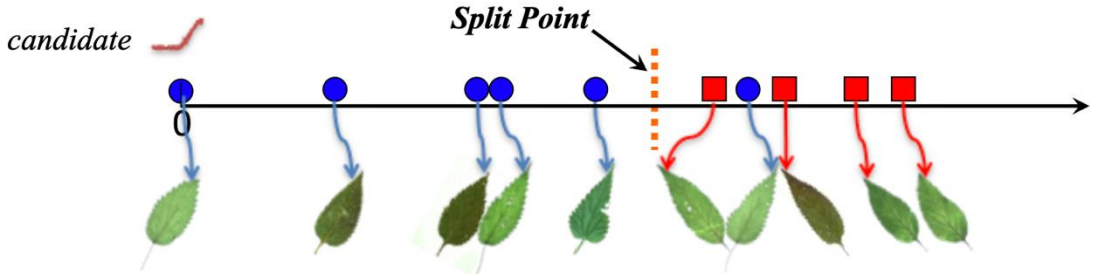


Entropy



- A TS dataset *D* consists of two classes, A and B.
- Given that the proportion of objects in class A is *p(A)* and the proportion of objects in class B is *p(B)*,
- The **Entropy** of D is: $I(D) = -p(A)log(p(A)) -p(B)log(p(B))$.
- Given a strategy that divides *D* into two subsets D_1 and D_2, the information remaining in the dataset after splitting is defined by the weighted average entropy of each subset.
- If the fraction of objects in D_1 is $f(D_1)$ and in D_2 is $f(D_2)$, the total entropy of *D* after splitting is $\hat{I}(D) = f(D_1)I(D_1) + f(D_2)I(D_2)$.

Information Gain

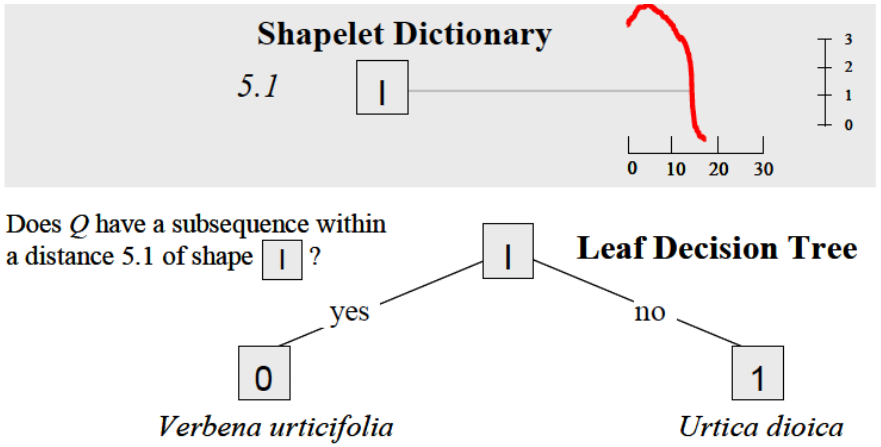


Split point distance from shapelet = 5.1

- Given a certain split strategy *sp* which divides *D* into two subsets D_1 and D_2, the entropy before and after splitting is $I(D)$ and $\hat{I}(D)$.
- The **information gain** for this splitting rule is:
- $Gain(sp) = I(D) - \hat{I}(D) =$

$$= I(D) - f(D_1)I(D_1) + f(D_2)I(D_2).$$

- We use the distance from *T* to a shapelet *S* as the splitting rule *sp*.



Problem

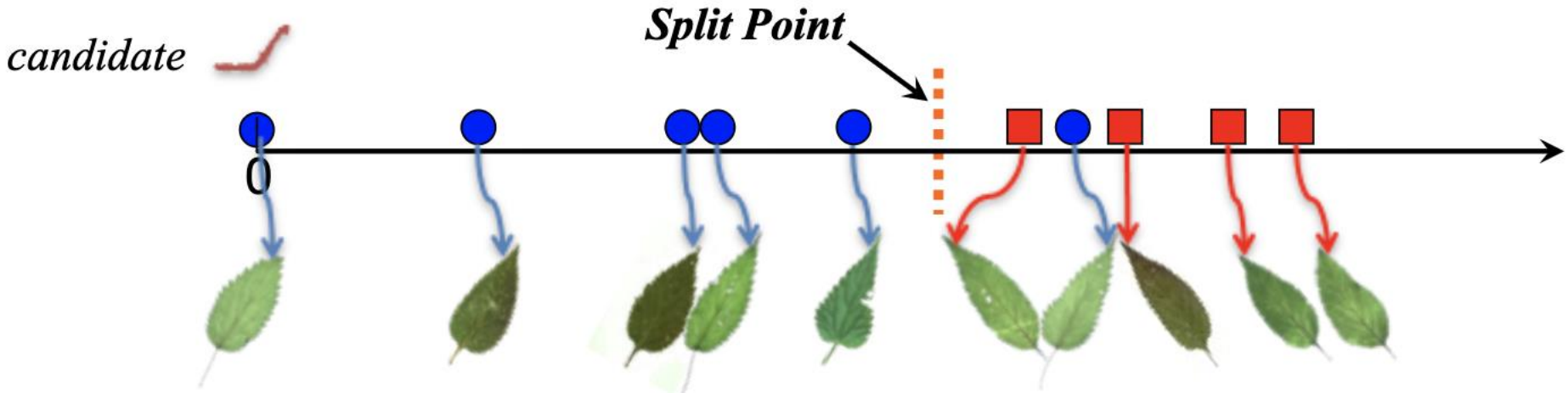
- The total number of candidate is $\sum_{l=MINLEN}^{MAXLEN} \sum_{T_i \in D} (|T_i| - l + 1)$

- For each candidate you have to compute the distance between this candidate and each training sample (space inefficiency)

- For instance
  - 200 instances with length 275
  - 7,480,200 shapelet candidates

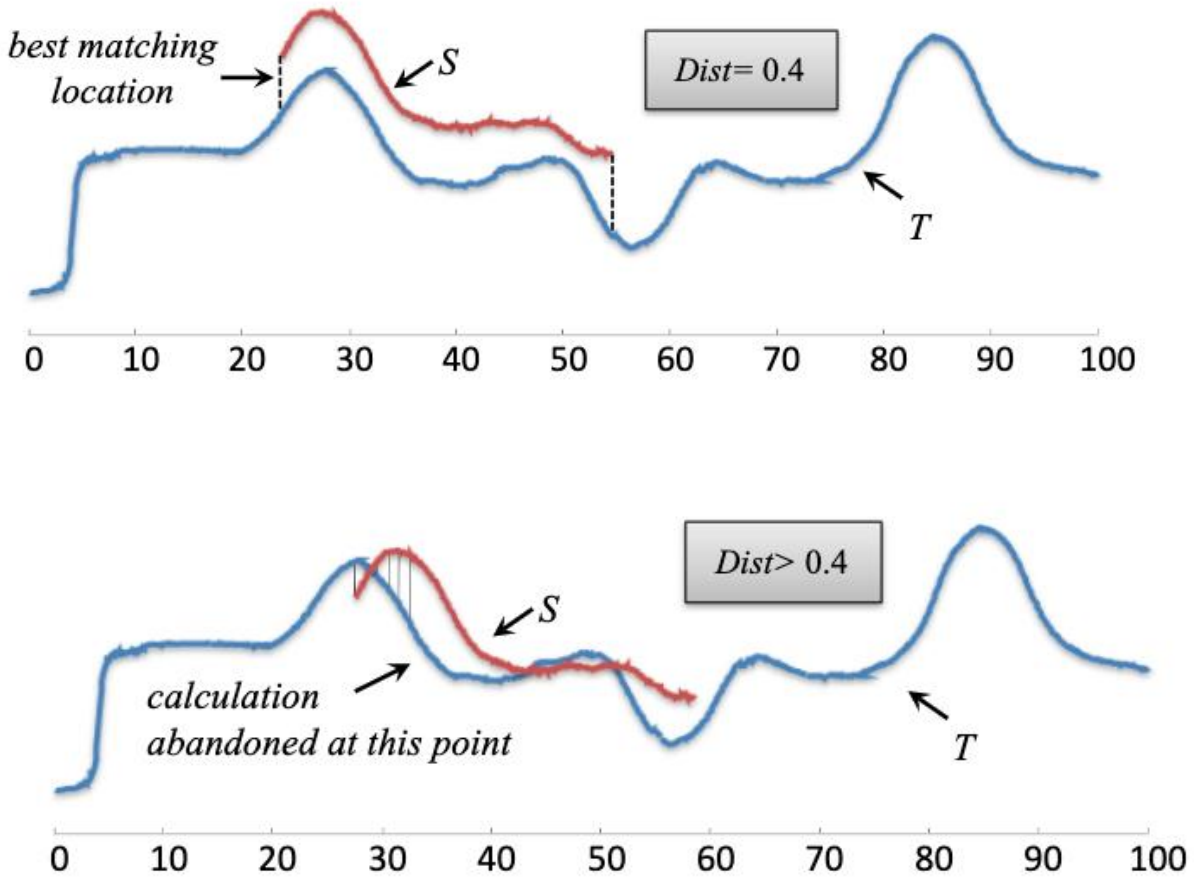
Speedup

- Distance calculations form TSs to shapelet candidates is expensive.
- Reduce the time in two ways
  - Distance Early Abandon: reducing the distance computation time between two TS
  - Admissible Entropy Pruning: reducing the number of distance calculations



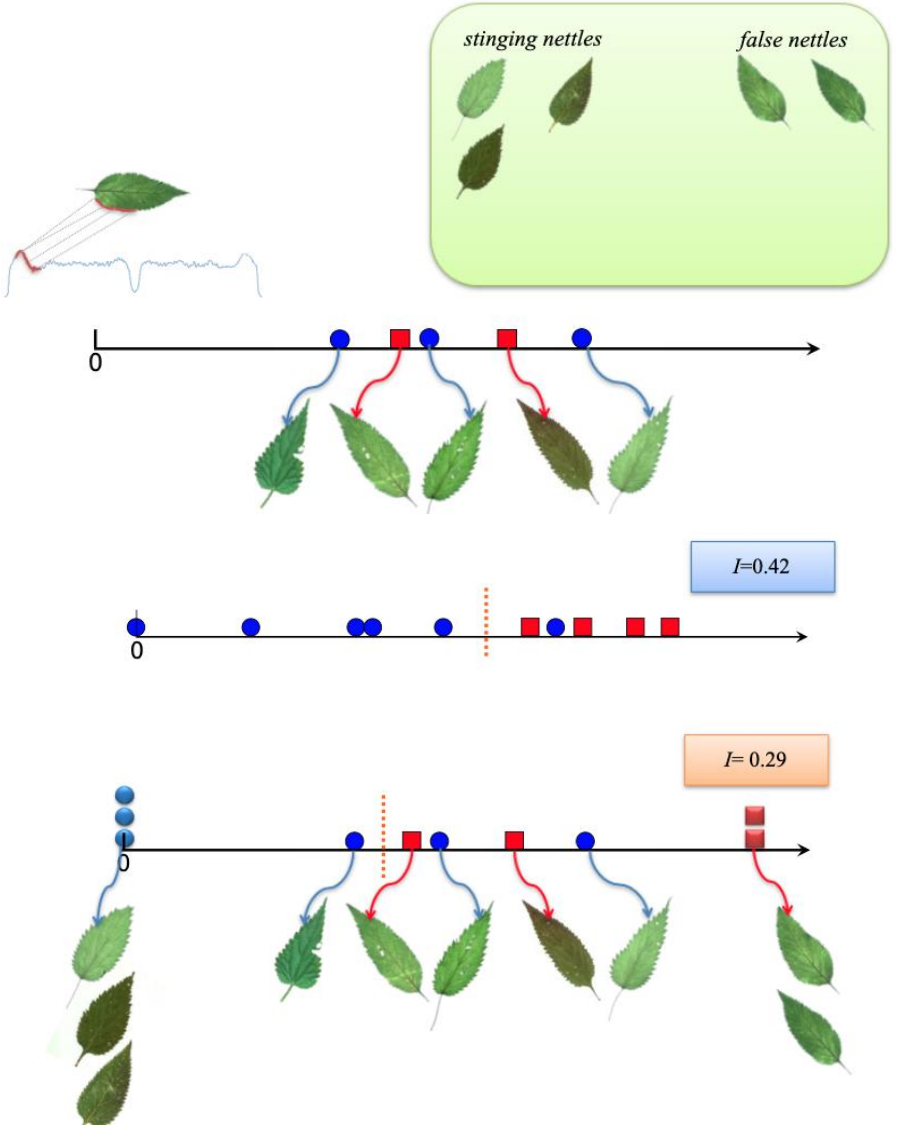
Distance Early Abandon

- We only need the minimum distance.
- Method:
  - Keep the best-so-far distance
  - Abandon the calculation if the partial current distance is larger than best-so-far.
  - We can avoid to compute the full distance for S if the partial one is greater than the best so far



Admissible Entropy Pruning

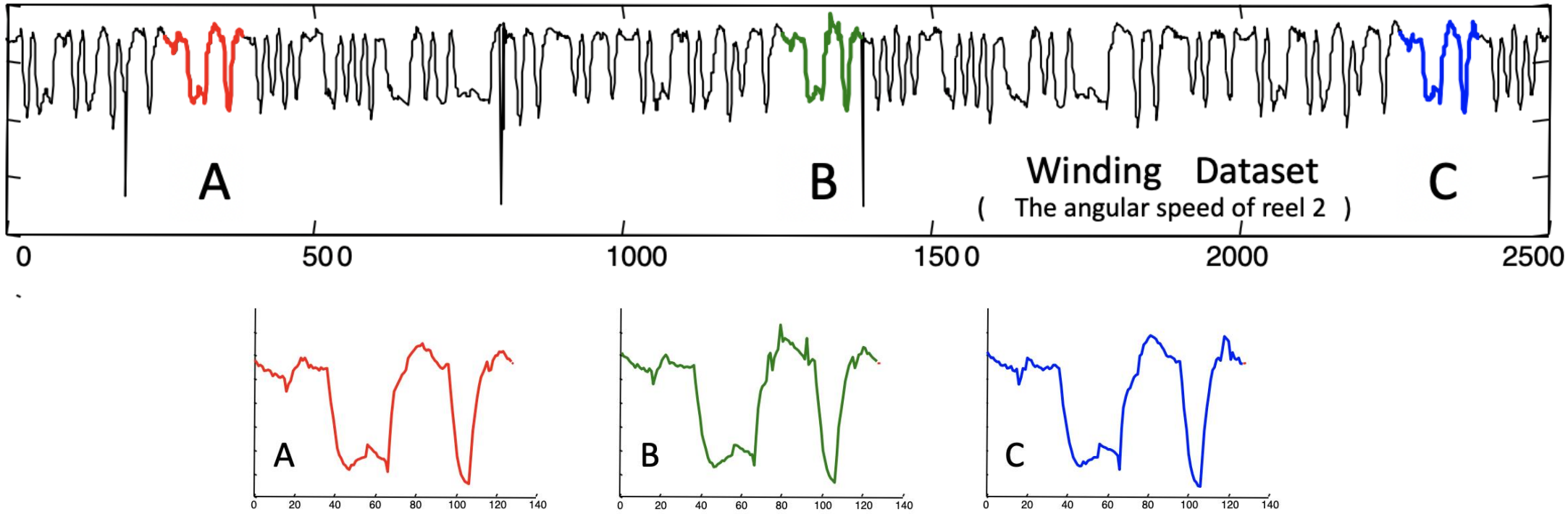
- We only need the best shapelet for each class
- For a candidate shapelet
  - We do not need to calculate the distance for each training sample
  - After calculating some training samples, if the **upper bound** of information gain (corresponding to the optimistic scenario) < best candidate shapelet
  - Stop calculation for that candidate and try next candidate



Motif

Time Series Motif Discovery

- Finding repeated patterns, i.e., pattern mining.
- Are there any repeated patterns, of length *m* in the TS?

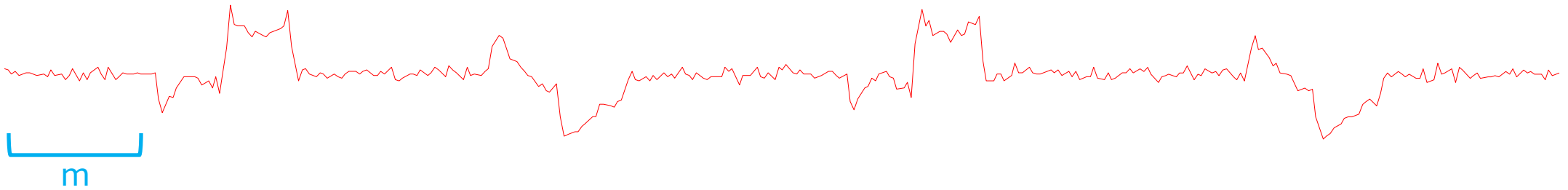


Why Find Motifs?

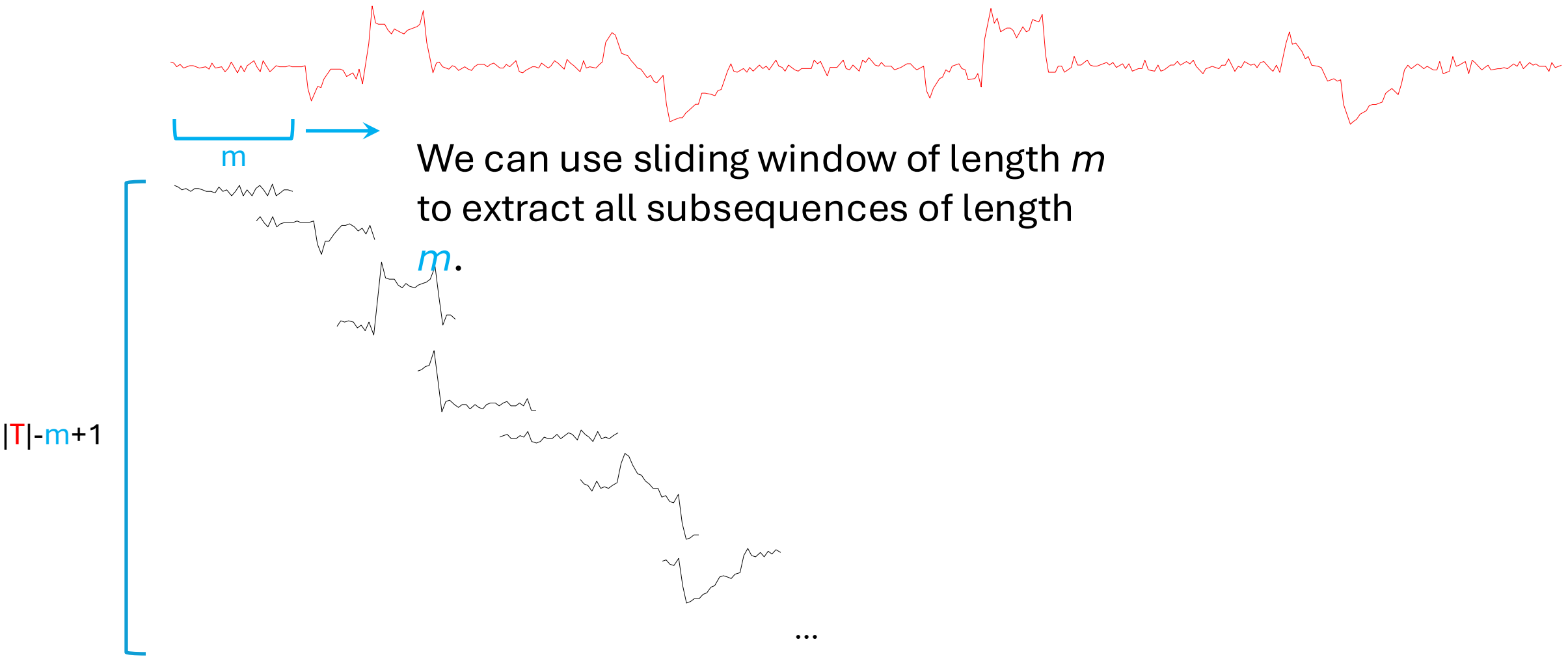
- Mining **association rules** in TS requires the discovery of motifs. These are referred to as primitive shapes and frequent patterns.
- Several **TS classifiers** work by constructing typical prototypes of each class. These prototypes may be considered motifs.
- Many **TS anomaly detection** algorithms consist of modeling normal behavior with a set of typical shapes (which we see as motifs), and detecting future patterns that are dissimilar to all typical shapes.

Matrix Profile

- The Matrix Profile (MP) is a data structure that annotates a TS and can be exploited for many purposed: e.g. efficient Motif Discovery.
- Given a time series, T and a desired subsequence length, m.

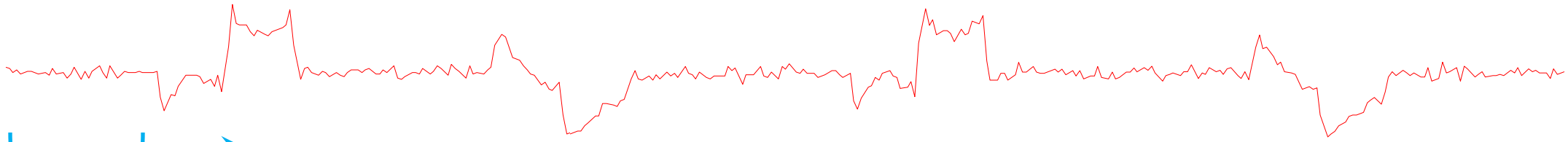


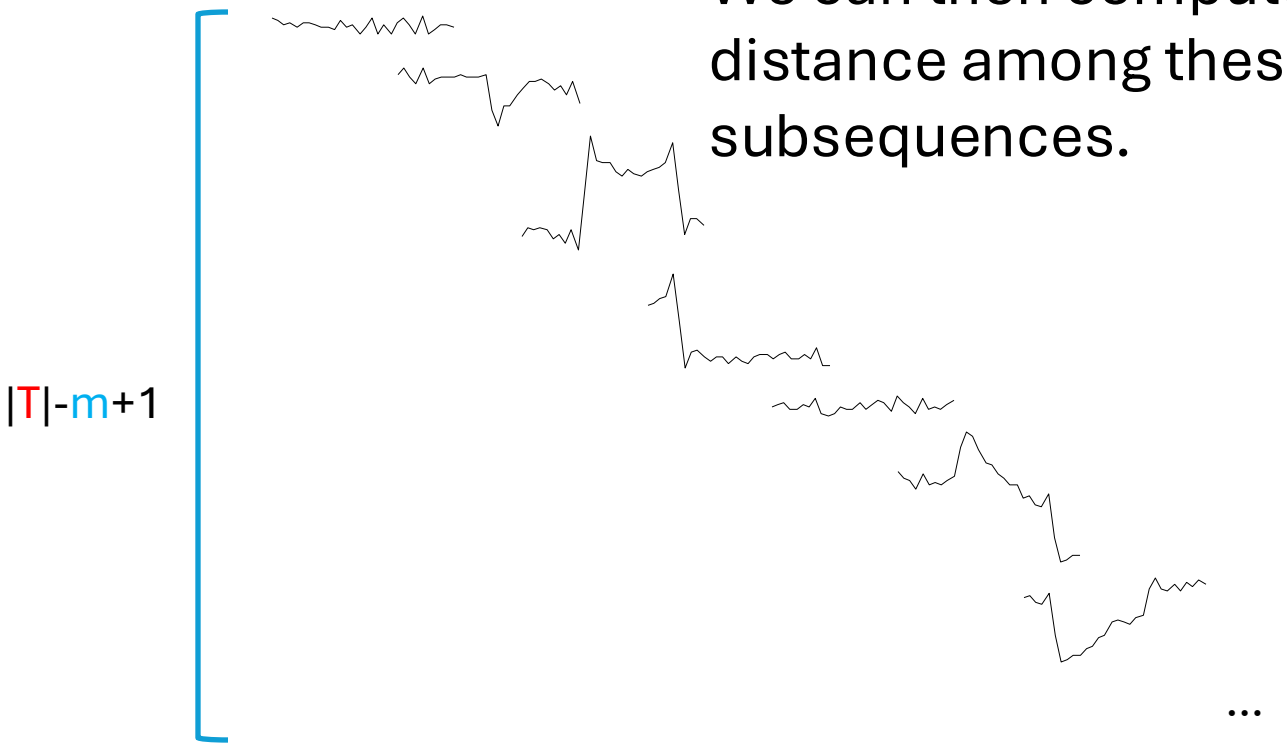
Matrix Profile



We can use sliding window of length *m* to extract all subsequences of length *m*.

Matrix Profile



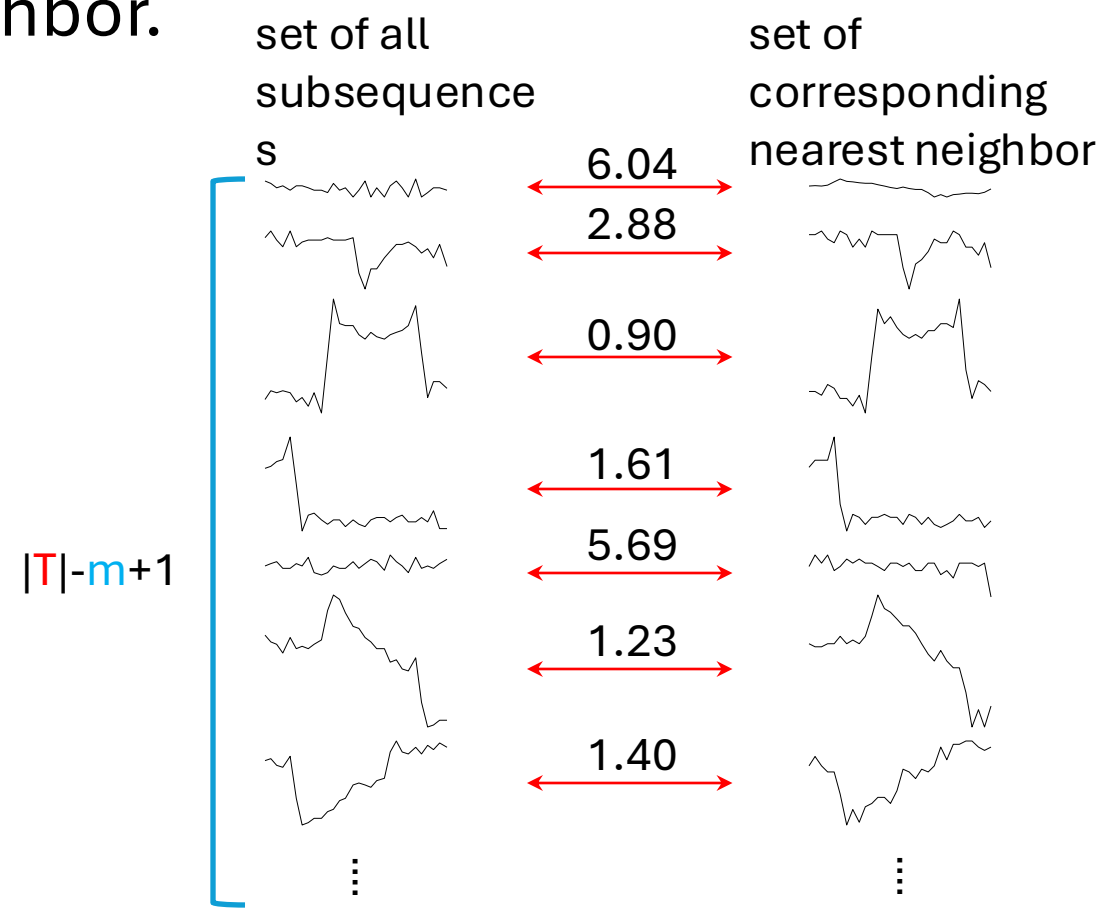


We can then compute the pairwise distance among these subsequences.

0	7.6952	7.7399	...
7.6952	0	7.7106	...
7.7399	7.7106	0	...
...	...	...	...

Matrix Profile

- For each subsequence we keep only the distance with the closest nearest neighbor.

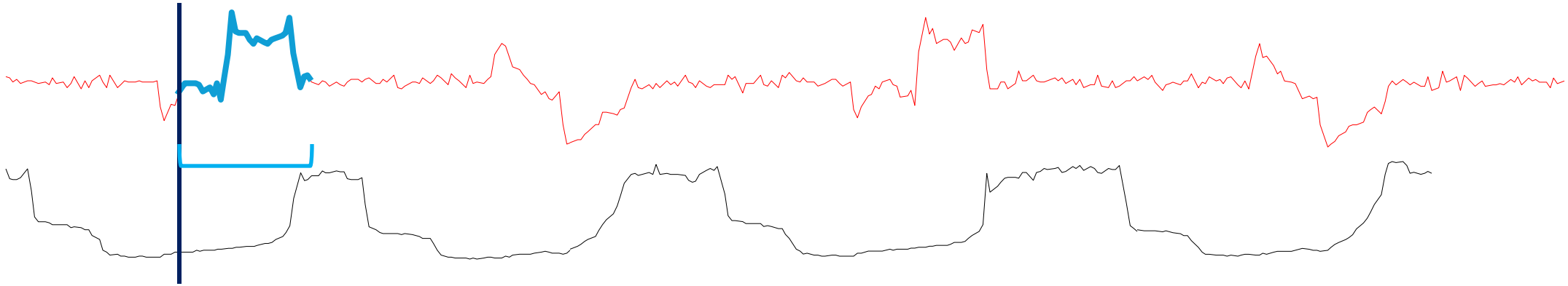


Matrix Profile

- The **distance** to the corresponding **nearest neighbor** of each subsequence can be stored in a vector called **matrix profile P**.

time series, T

matrix profile, P



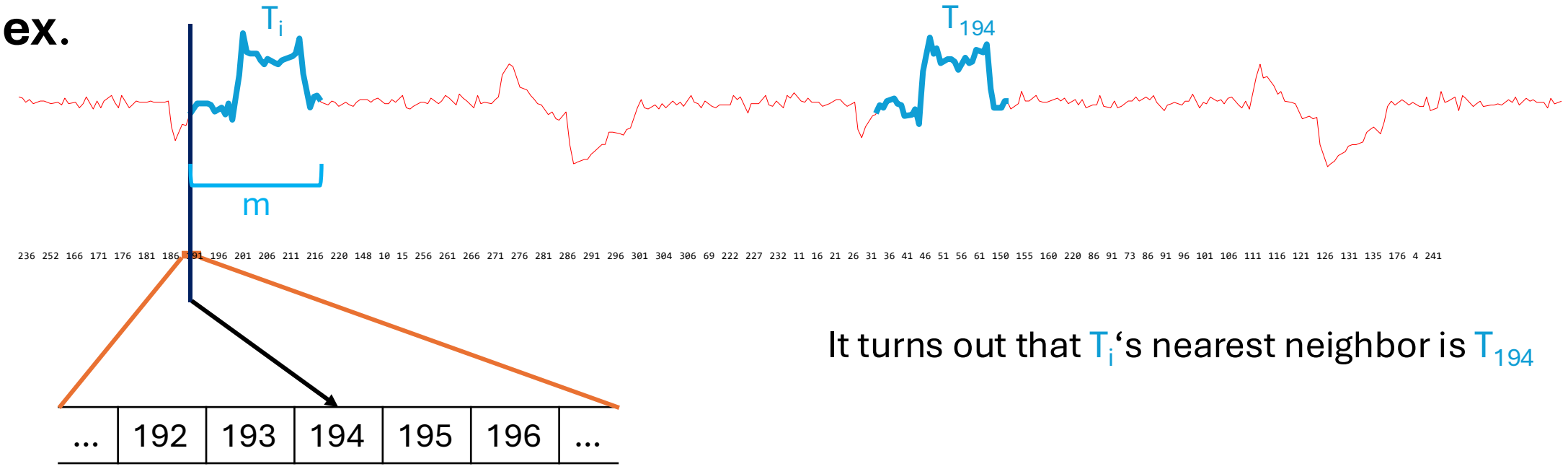
The matrix profile value at location *i* is the distance between T_i and its nearest neighbor

Matrix Profile

- The **index of corresponding nearest neighbor** of each subsequence is also stored in a vector called **matrix profile index.**

time series, T

matrix profile index, I

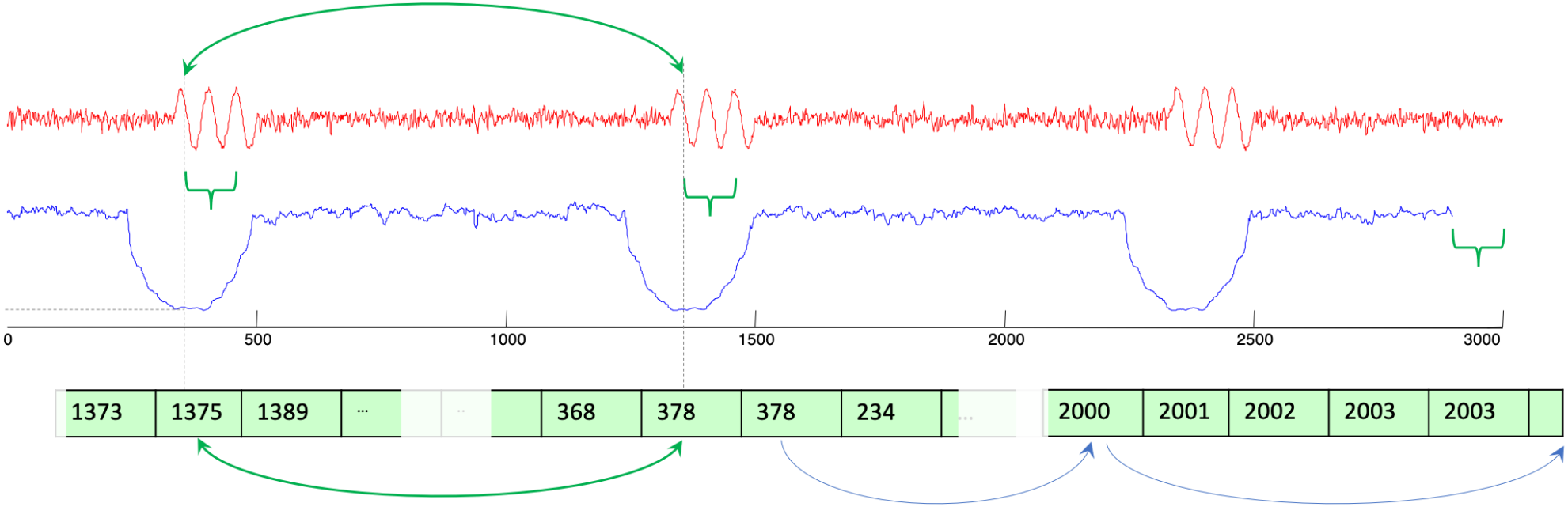


It turns out that T_i's nearest neighbor is T_{194}

The matrix profile value at location *i* is the distance between T_i and its nearest neighbor

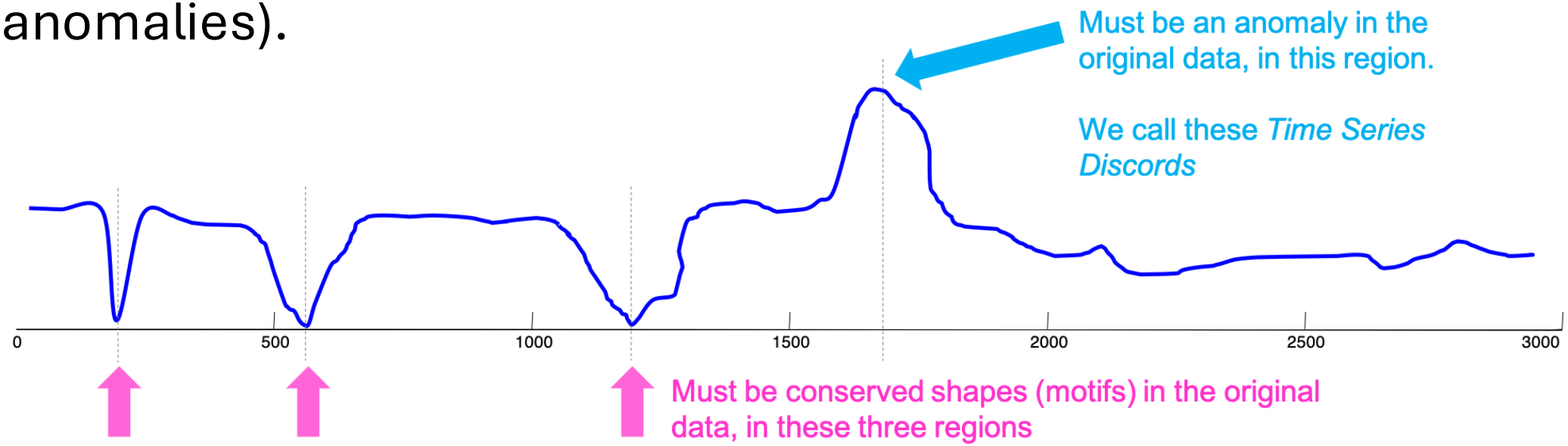
Matrix Profile

- The MP index allows to find the nearest neighbor to any subsequence in constant time.
- Note that the pointers in the matrix profile index are not necessarily symmetric.
- If A points to B, then B may or may not point to A
- The classic TS motif: the two smallest values in the MP must have the same value, and their pointers must be mutual.



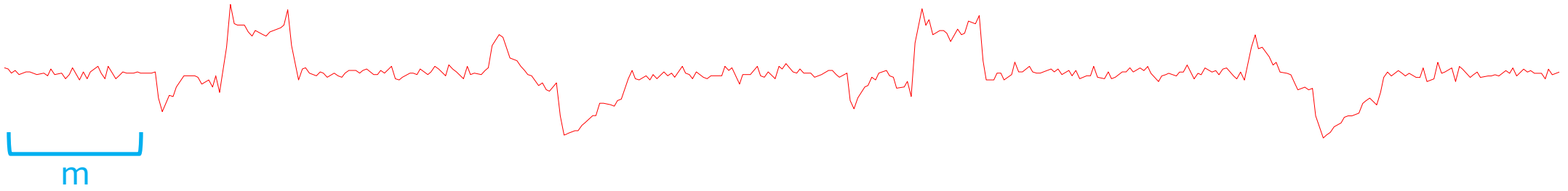
How to “read” a Matrix Profile

- For relatively low values, you know that the subsequence in the original TS must have (at least one) relatively similar subsequence elsewhere in the data (such regions are “motifs”)
- For relatively high values, you know that the subsequence in the original TS must be unique in its shape (such areas are anomalies).



How to Compute Matrix Profile?

- Given a time series, T and a desired subsequence length, m.



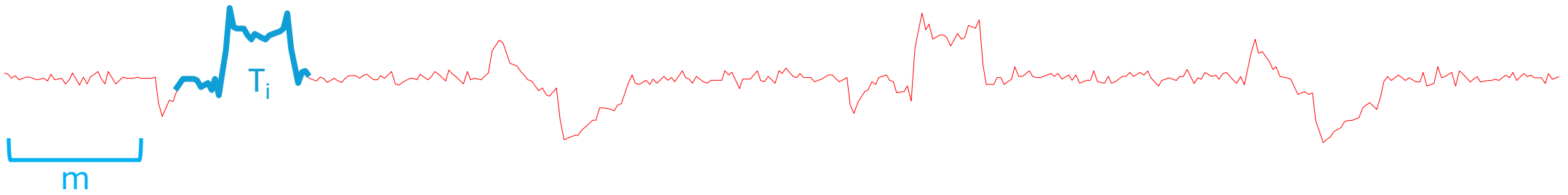
inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

Matrix profile is initialized as inf vector

This is just a toy example, so the values and the vector length does not fit the time series shown above

How to Compute Matrix Profile?

- Given a time series, T and a desired subsequence length, m.

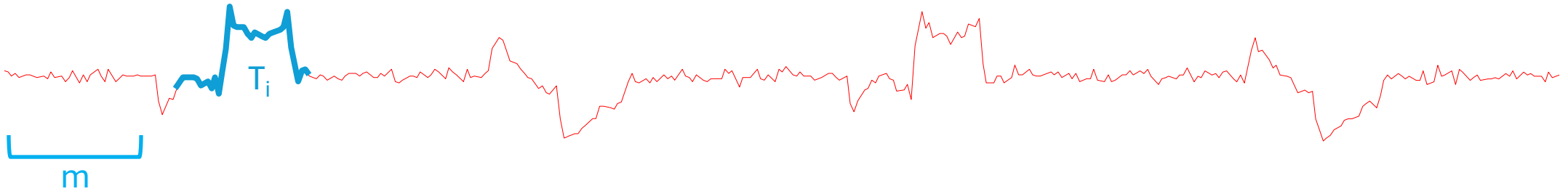


inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

At the first iteration, a subsequence T_i is randomly selected from T

How to Compute Matrix Profile?

- Given a time series, T and a desired subsequence length, m.



inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

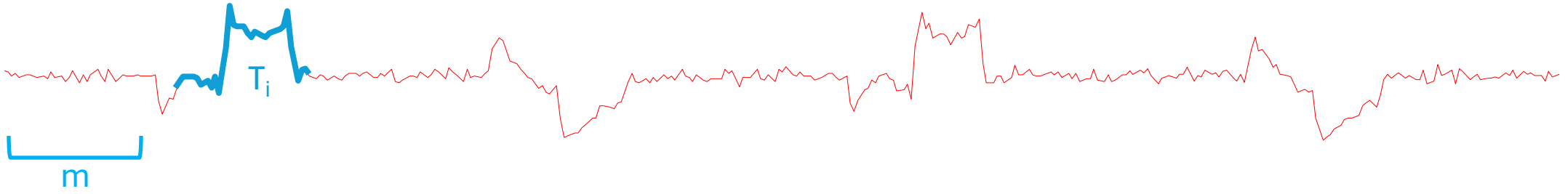
We compute the distances between T_i and every subsequences from T (time complexity = O(|T|log(|T|)))
We then put the distances in a vector based on the position of the subsequences

3	2	0	5	3	4	5	1	2	9	8	4	2	3	4	8	6	2	1

The distance between T_i and T_1 (first subsequence) is 3

How to Compute Matrix Profile?

- Given a time series, T and a desired subsequence length, m.



inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

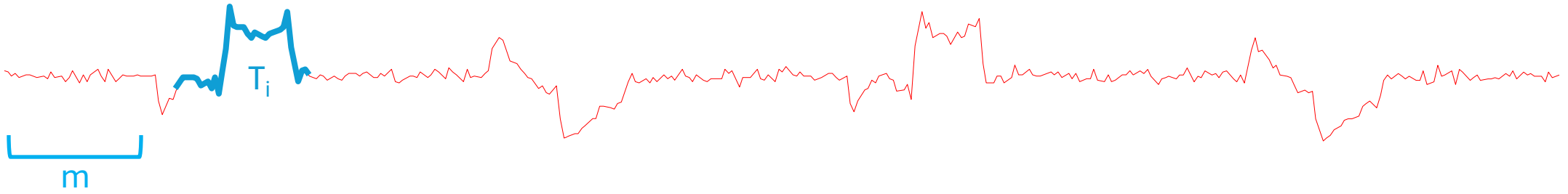
We compute the distances between T_i and every subsequences from T (time complexity = O(|T|log(|T|)))
We them put the distances in a vector based on the position of the subsequences

3	2	0	5	3	4	5	1	2	9	8	4	2	3	4	8	6	2	1

Let say T_i happen to be the third subsequences,
therefore the third value in the distance vector is 0

How to Compute Matrix Profile?

- Given a time series, T and a desired subsequence length, m.



inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

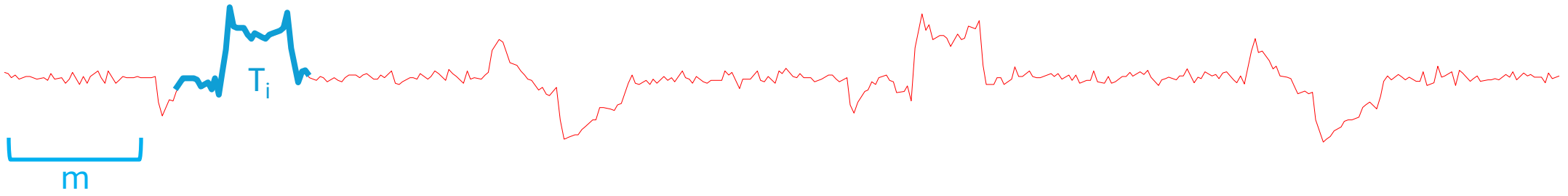
min

Matrix profile is updated by apply elementwise minimum to these two vectors

3	2	0	5	3	4	5	1	2	9	8	4	2	3	4	8	6	2	1

How to Compute Matrix Profile?

- Given a time series, T and a desired subsequence length, m.



3	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf	inf

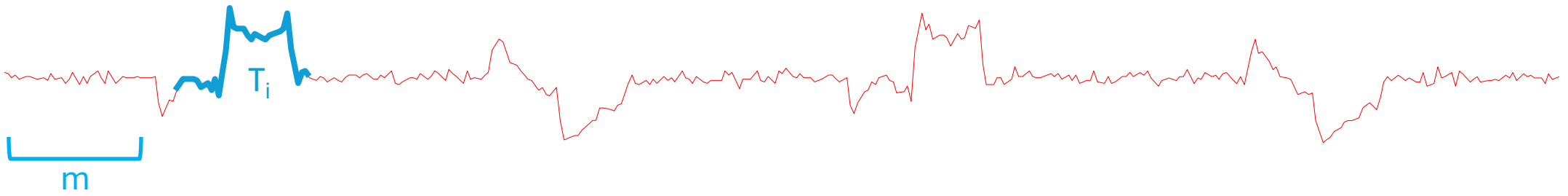
min

Matrix profile is updated by apply elementwise minimum to these two vectors

3	2	0	5	3	4	5	1	2	9	8	4	2	3	4	8	6	2	1

How to Compute Matrix Profile?

- Given a time series, T and a desired subsequence length, m.



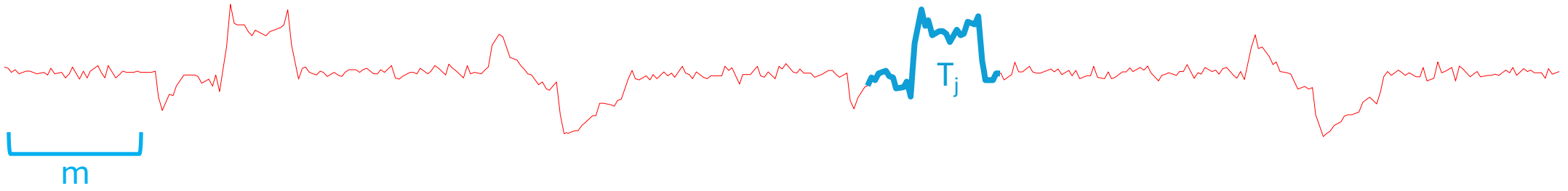
3	2	inf	5	3	4	5	1	2	9	8	4	2	3	4	8	6	2	1

After we finish to update matrix profile for the first iteration

3	2	0	5	3	4	5	1	2	9	8	4	2	3	4	8	6	2	1

How to Compute Matrix Profile?

- Given a time series, T and a desired subsequence length, m.

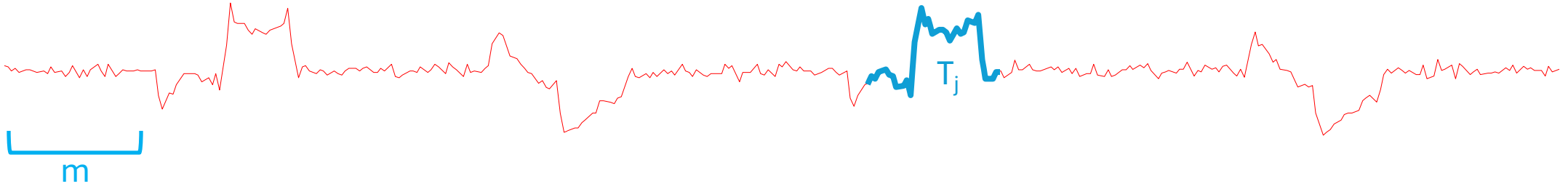


3	2	inf	5	3	4	5	1	2	9	8	4	2	3	4	8	6	2	1

In the second iteration, we randomly select another subsequence T_j and it happens to be the 12th subsequences

How to Compute Matrix Profile?

- Given a time series, T and a desired subsequence length, m.



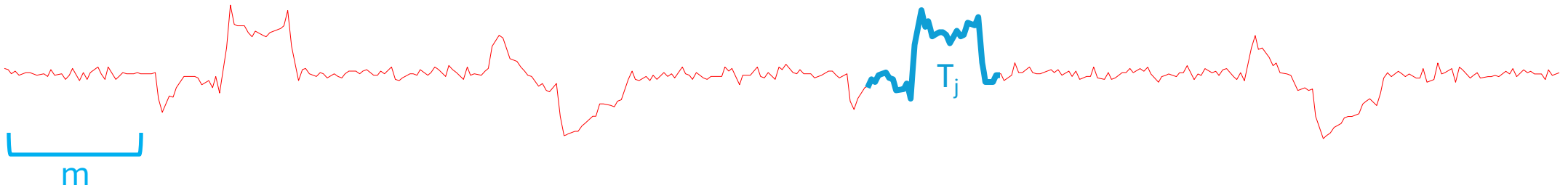
3	2	inf	5	3	4	5	1	2	9	8	4	2	3	4	8	6	2	1

Once again, we compute the distance between T_j and every subsequences of T

2	3	1	4	4	3	6	2	1	5	8	0	2	3	5	9	4	2	2

How to Compute Matrix Profile?

- Given a time series, T and a desired subsequence length, m.



3	2	inf	5	3	4	5	1	2	9	8	4	2	3	4	8	6	2	1

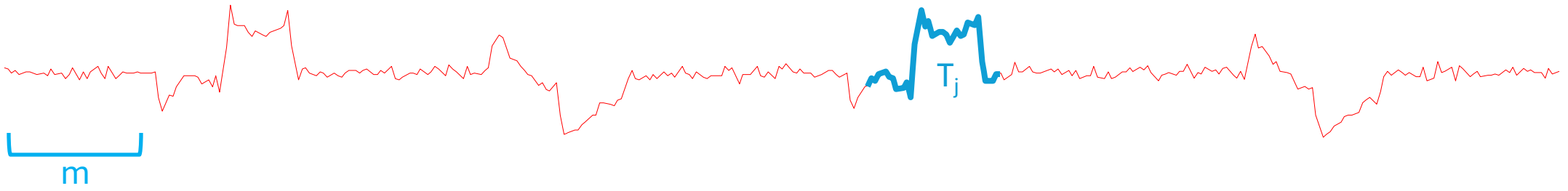
min

The same elementwise minimum

2	3	1	4	4	3	6	2	1	5	8	0	2	3	5	9	4	2	2

How to Compute Matrix Profile?

- Given a time series, T and a desired subsequence length, m.



2	2	inf	5	3	4	5	1	2	9	8	4	2	3	4	8	6	2	1

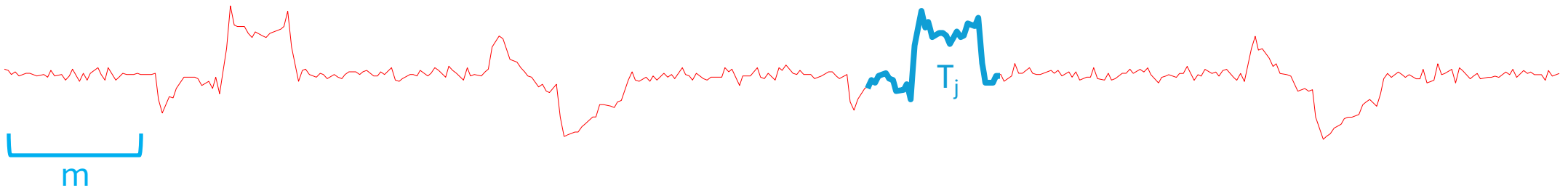
min

The same elementwise minimum

2	3	1	4	4	3	6	2	1	5	8	0	2	3	5	9	4	2	2

How to Compute Matrix Profile?

- Given a time series, T and a desired subsequence length, m.



2	2	inf	5	3	4	5	1	2	9	8	4	2	3	4	8	6	2	1

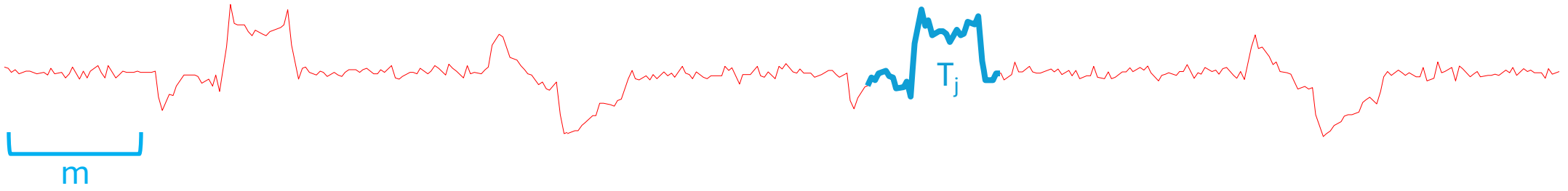
min

The same elementwise minimum

2	3	1	4	4	3	6	2	1	5	8	0	2	3	5	9	4	2	2

How to Compute Matrix Profile?

- Given a time series, T and a desired subsequence length, m.



2	2	1	5	3	4	5	1	2	9	8	4	2	3	4	8	6	2	1

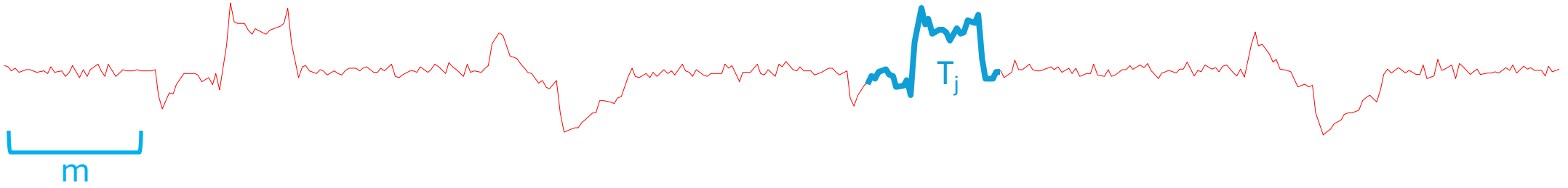
min

The same elementwise minimum

2	3	1	4	4	3	6	2	1	5	8	0	2	3	5	9	4	2	2

How to Compute Matrix Profile?

- Given a time series, T and a desired subsequence length, m.



2	2	1	5	3	4	5	1	2	9	8	4	2	3	4	8	6	2	1

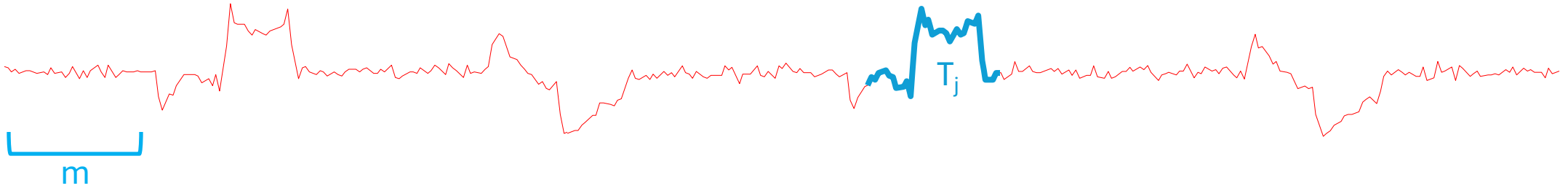
min

2	3	1	4	4	3	6	2	1	5	8	0	2	3	5	9	4	2	2

We repeat the two steps (distance computation and update) until we have used every subsequences

How to Compute Matrix Profile?

- Given a time series, T and a desired subsequence length, m.



2	2	1	5	3	4	5	1	2	9	8	4	2	3	4	8	6	2	1

min

2	3	1	4	4	3	6	2	1	5	8	0	2	3	5	9	4	2	2

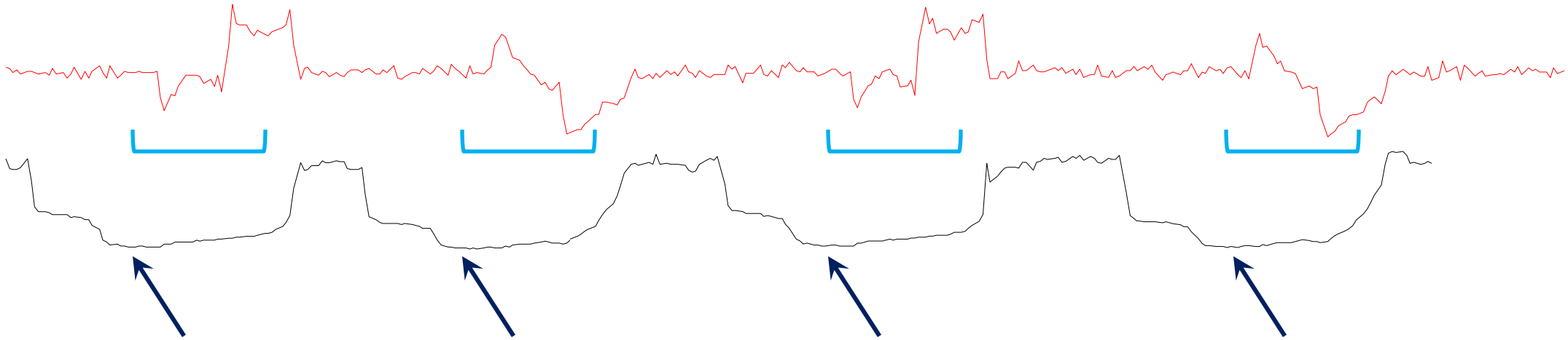
There are |T| subsequences and the distance computation is O(|T|log(|T|))

The overall time complexity is $O(|T|^2 log(|T|))$

Motif Discovery From Matrix Profile

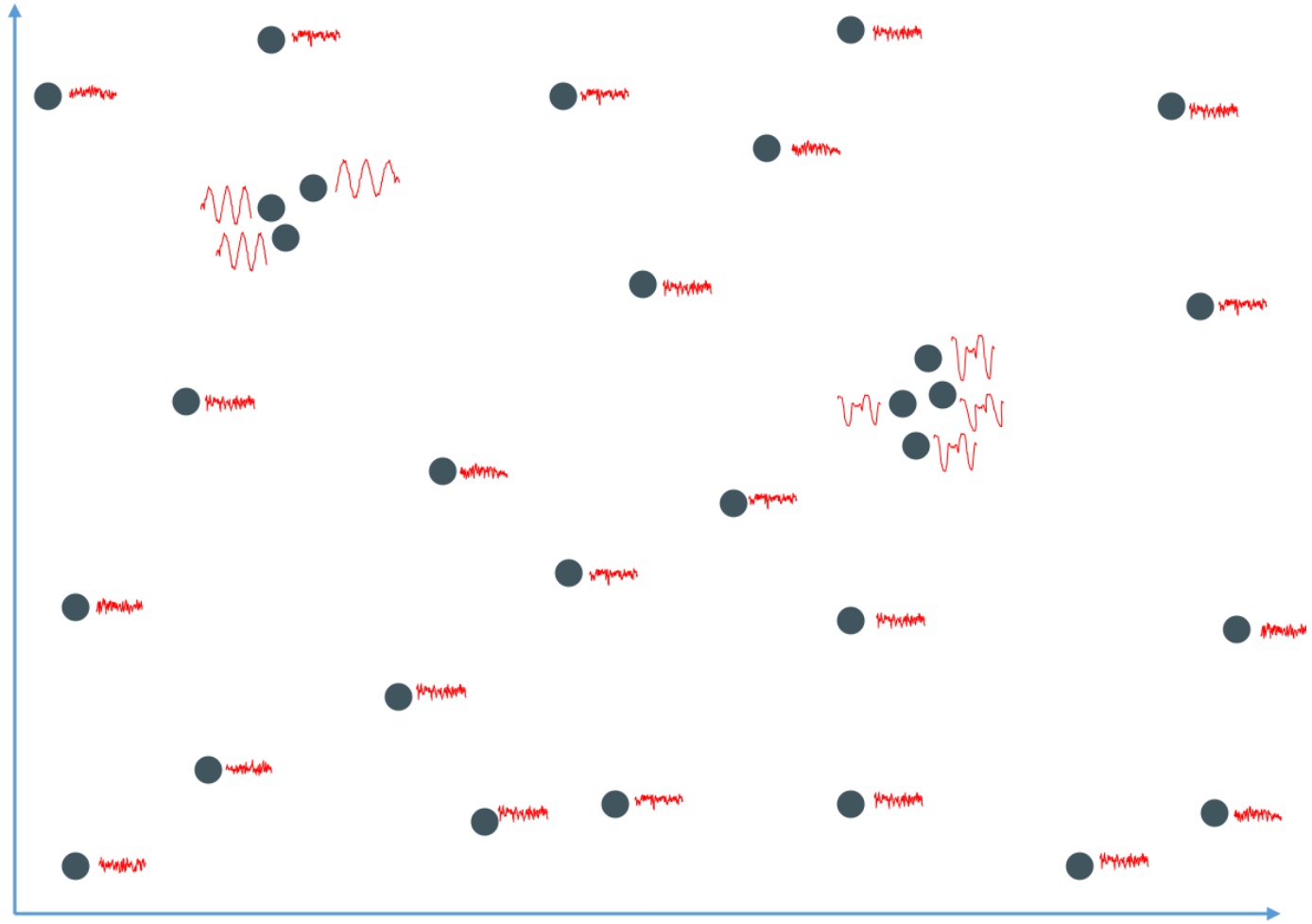
time series, T

matrix profile, P

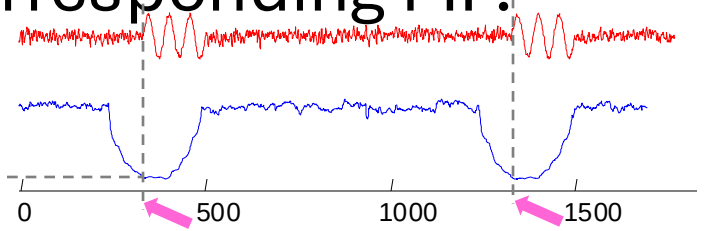


Local minimums are corresponding to motifs

Motif Discovery From Matrix Profile

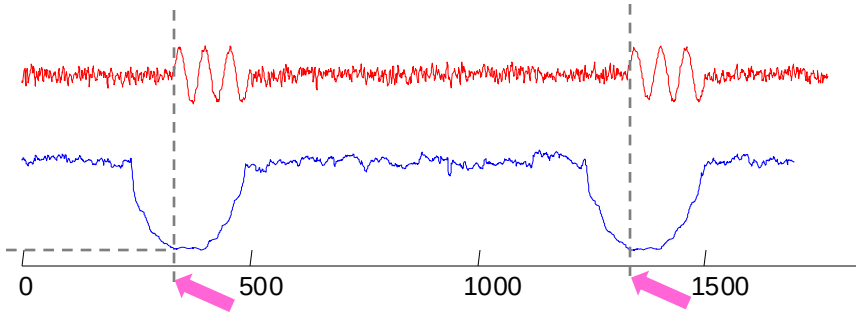


- It is sometime useful to think of time series subsequences as points in m-dimensional space.
- In this view, dense regions in the m-dimensional space correspond to regions of the time series that have a low corresponding MP.

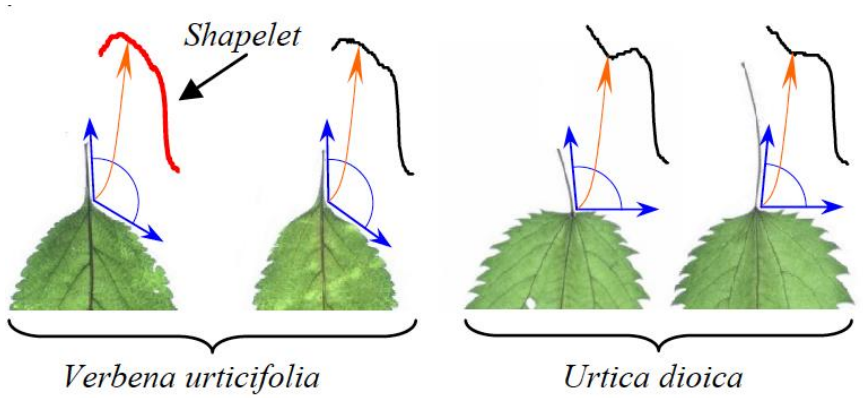


Motif/Shapelet Summary

- A **motif** is a repeated pattern/subsequence in a given TS.



- A **shapelet** is a pattern/subsequence which is maximally representative of a class with respect to a given dataset of TSs.



References

- Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View that Includes Motifs, Discords and Shapelets. Chin-Chia Michael Yeh et al. 1997
- Time Series Shapelets: A New Primitive for Data Mining. Lexiang Ye and Eamonn Keogh. 2016.
- Josif Grabocka, Nicolas Schilling, Martin Wistuba, Lars Schmidt-Thieme (2014): Learning Time-Series Shapelets, in Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2014

Matrix Profile I: All Pairs Similarity Joins for Time Series:
A Unifying View that Includes Motifs, Discords and Shapelets

Time Series Shapelets: A New Primitive for Data Mining