

DATA MINING 2

Time Series Classification

Riccardo Guidotti

a.a. 2020/2021

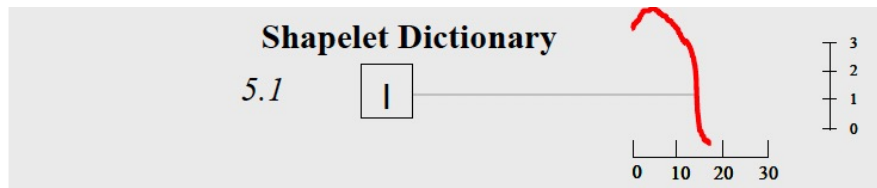
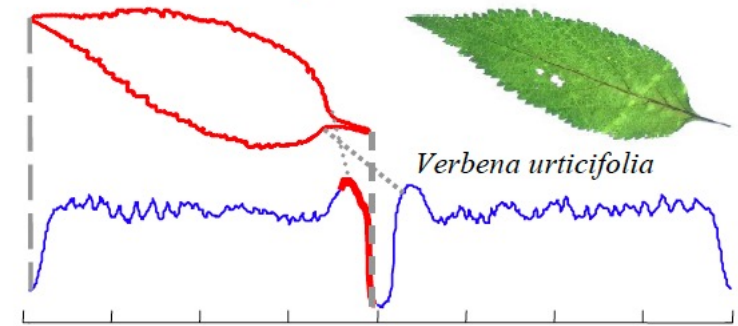
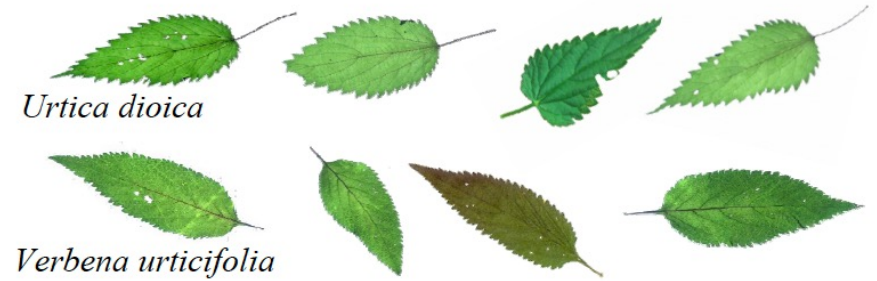


Time Series Classification

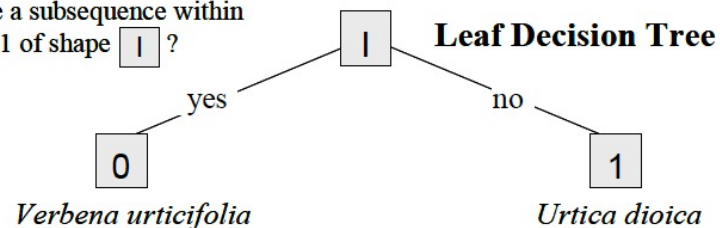
- Given a set X of n time series, $X = \{x_1, x_2, \dots, x_n\}$, each time series has m ordered values $x_i = \langle x_{t1}, x_{t2}, \dots, x_{tm} \rangle$ and a class value c_i .
- The objective is to find a function f that maps from the space of possible time series to the space of possible class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

Shapelet-based Classification

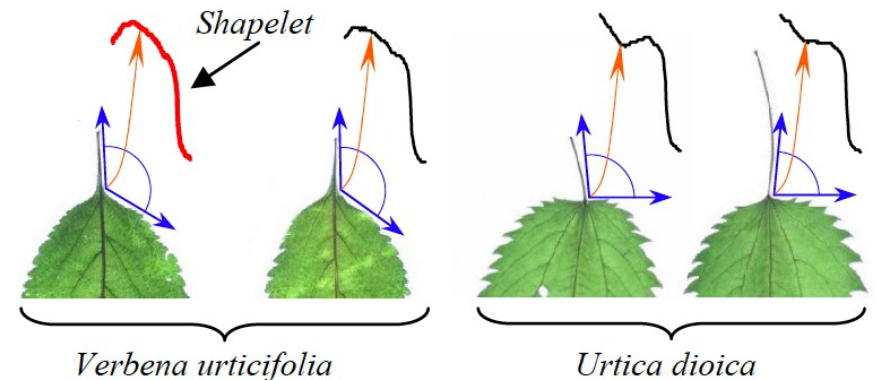
1. Represent a TS as a vector of distances with representative subsequences, namely shapelets.
2. Use it to as input for machine learning classifiers.



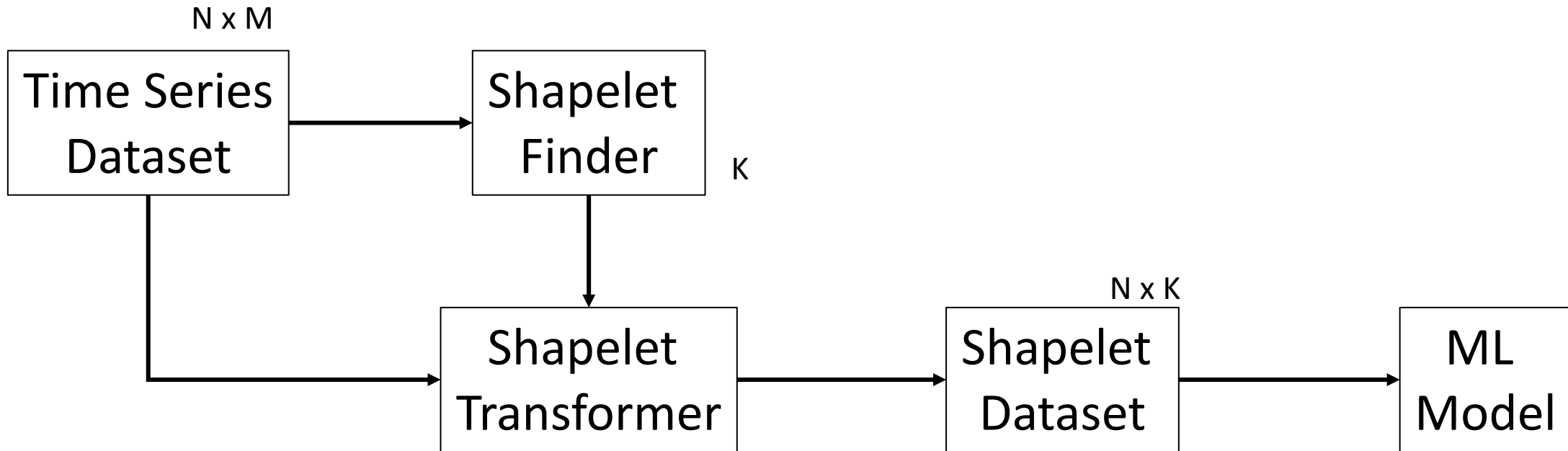
Does Q have a subsequence within a distance 5.1 of shape 1?



3.2	8.7
1.4	7.9
6.7	4.2
9.2	3.4

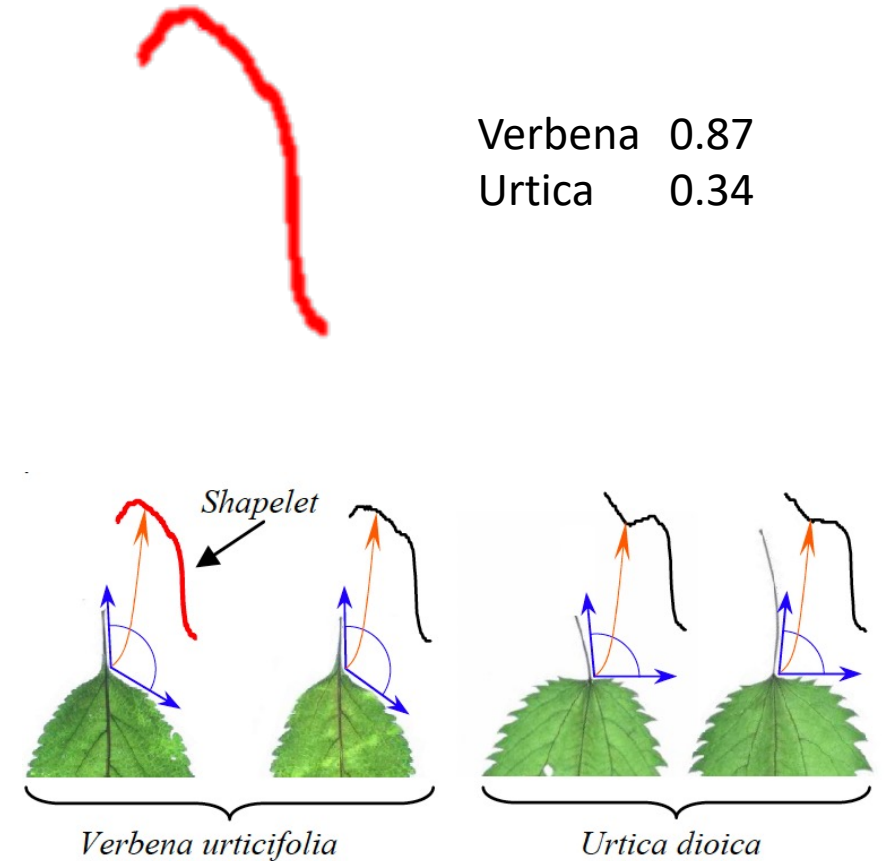


Shaplet-based Classifiers

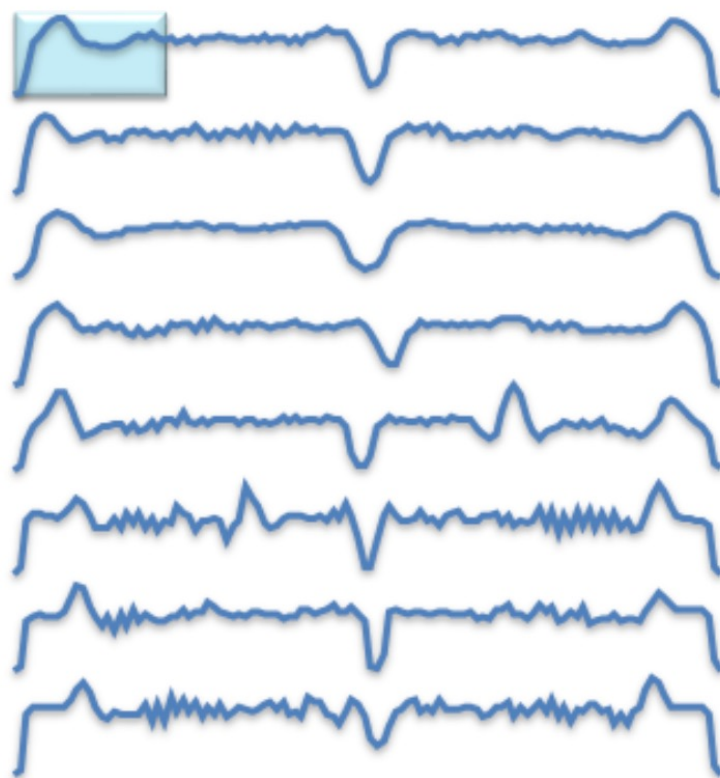


Time Series Shapelets

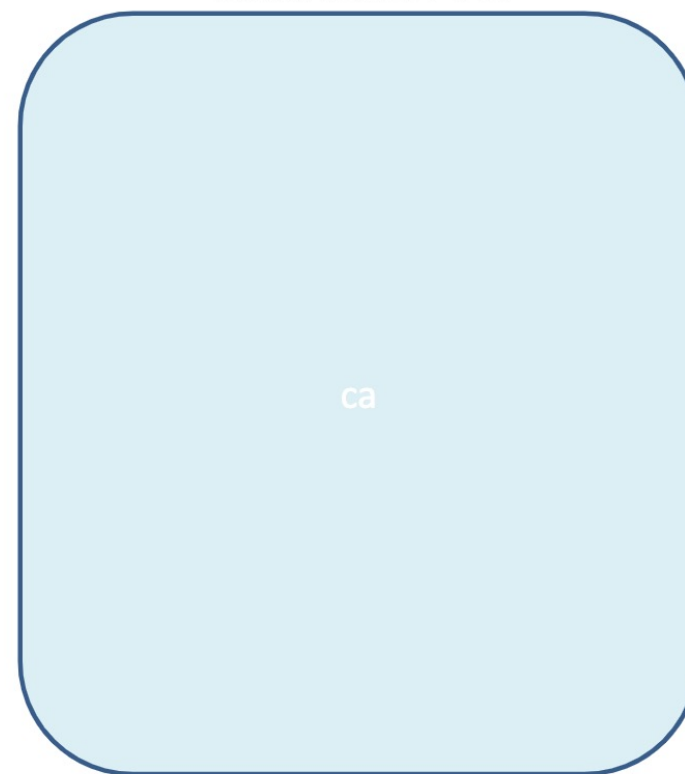
- Shapelets are TS subsequences which are maximally representative of a class.
- Shapelets can provide interpretable results, which may help domain practitioners better understand their data.
- Shapelets can be significantly more accurate/robust because they are *local features*, whereas most other state-of-the-art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



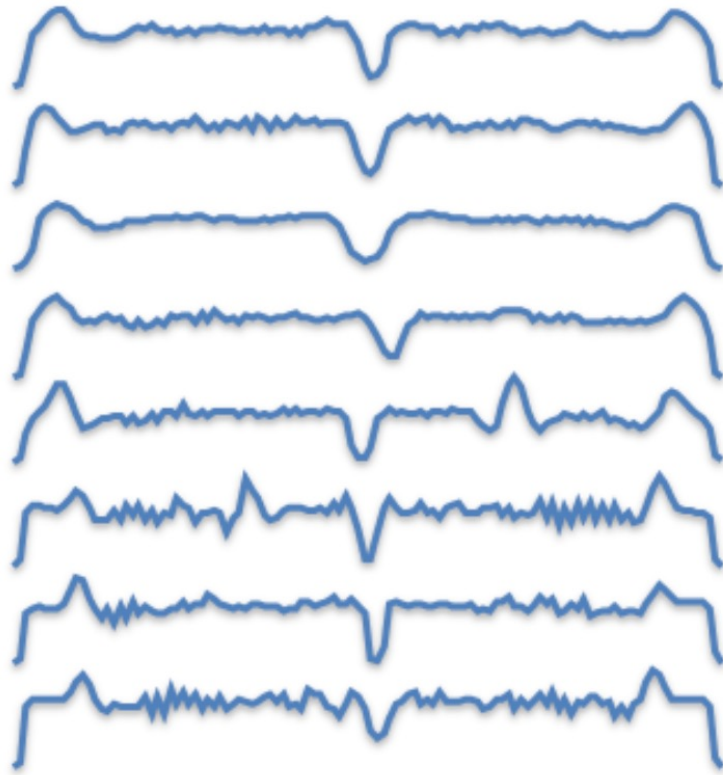
Extract Subsequences of all Possible Lengths



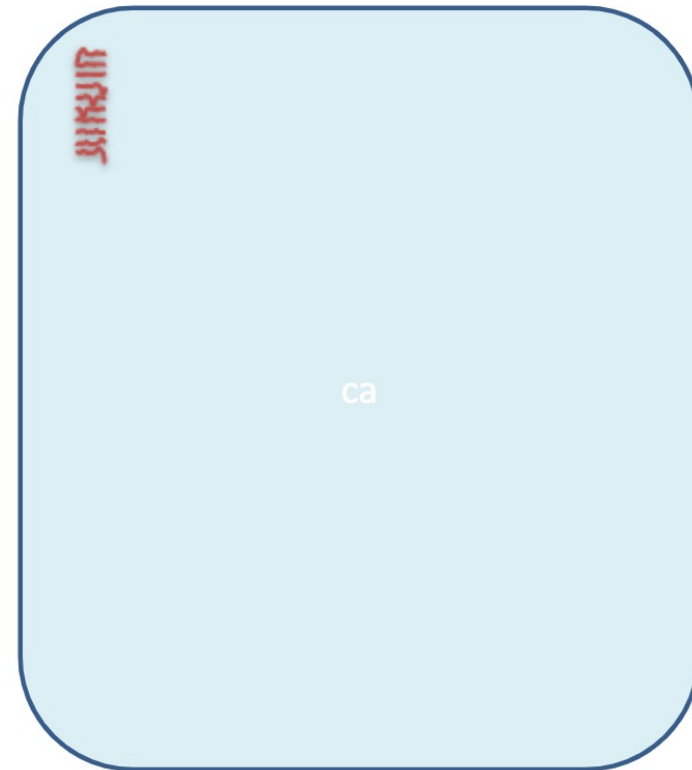
Candidates Pool



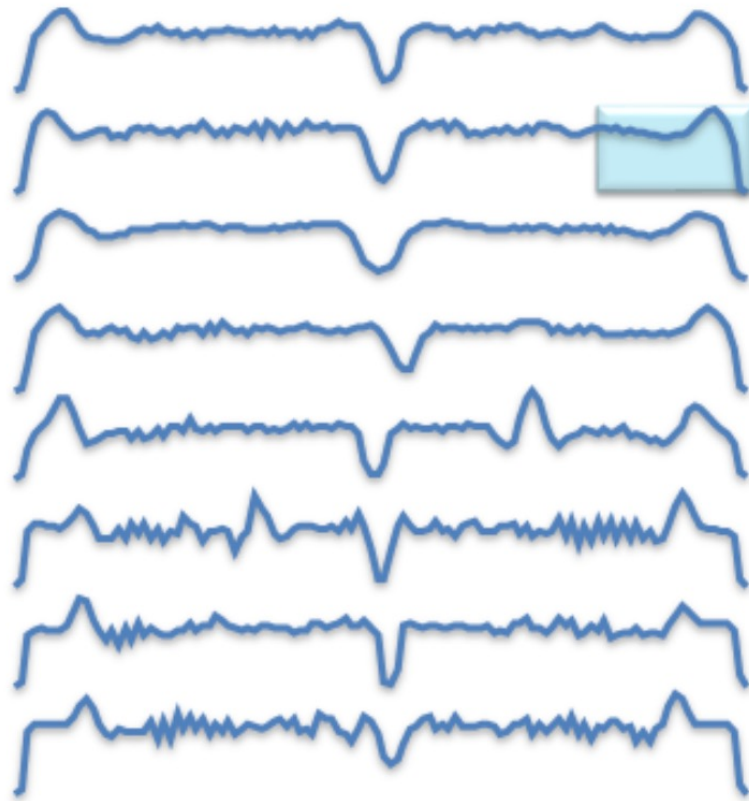
Extract Subsequences of all Possible Lengths



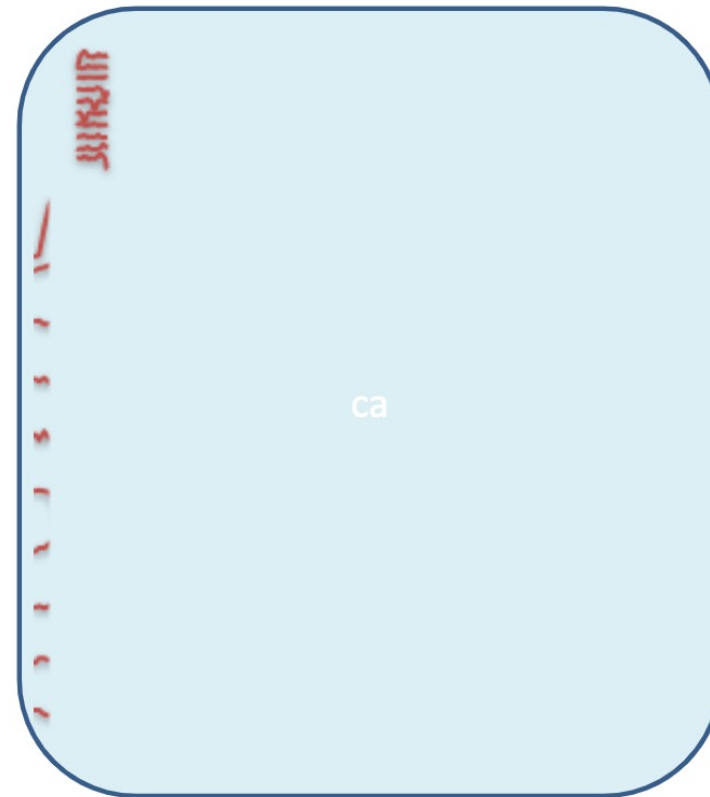
Candidates Pool



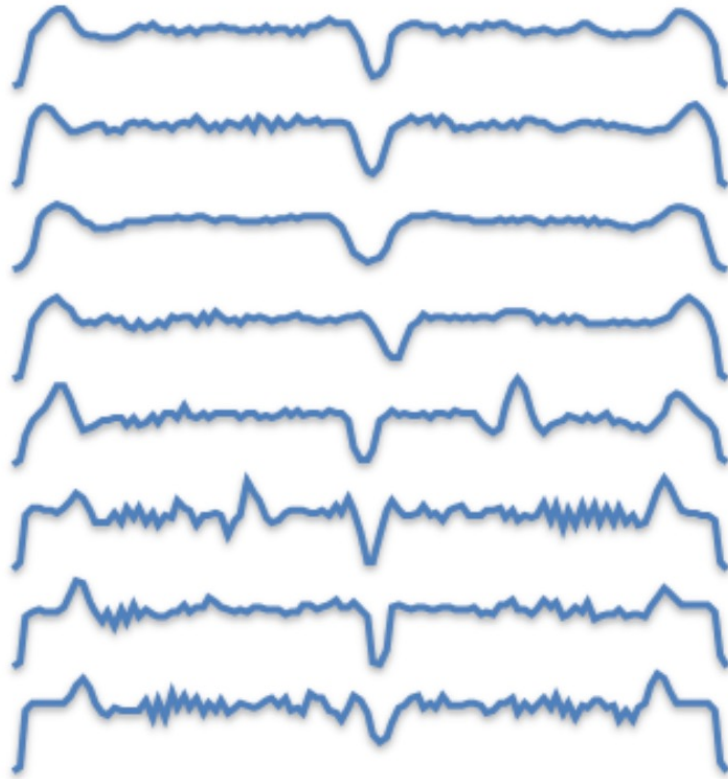
Extract Subsequences of all Possible Lengths



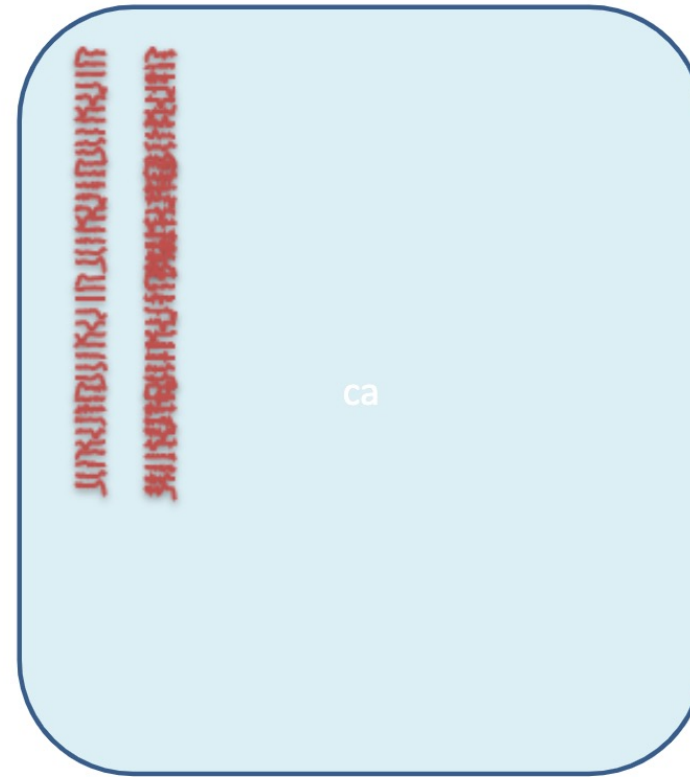
Candidates Pool



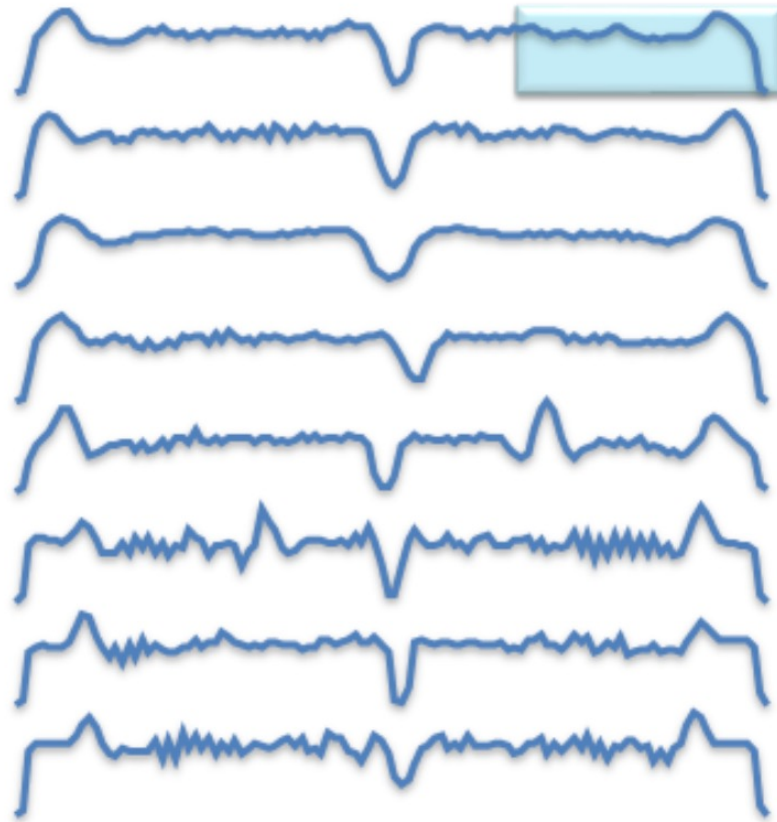
Extract Subsequences of all Possible Lengths



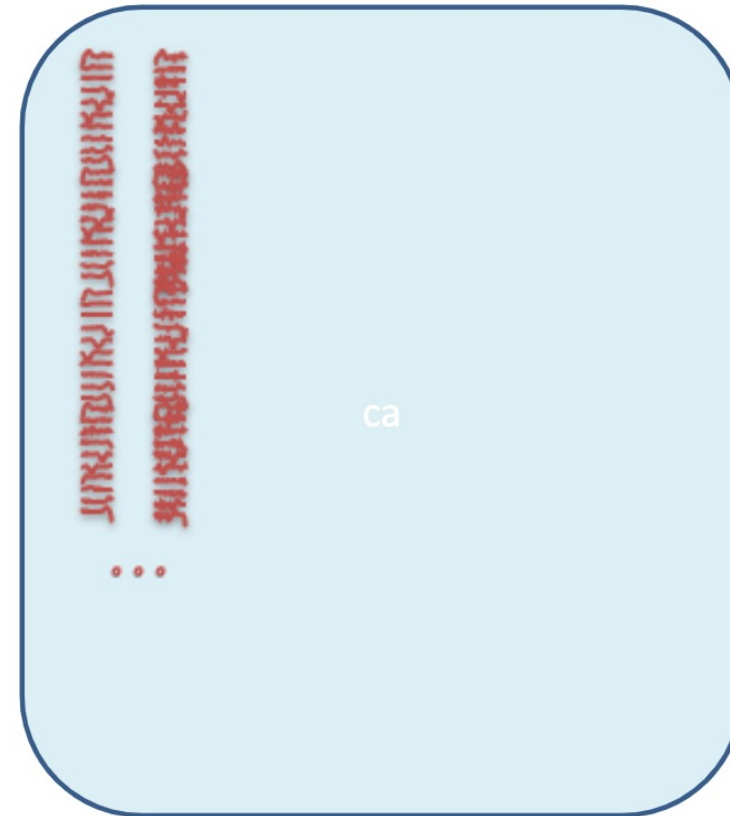
Candidates Pool



Extract Subsequences of all Possible Lengths

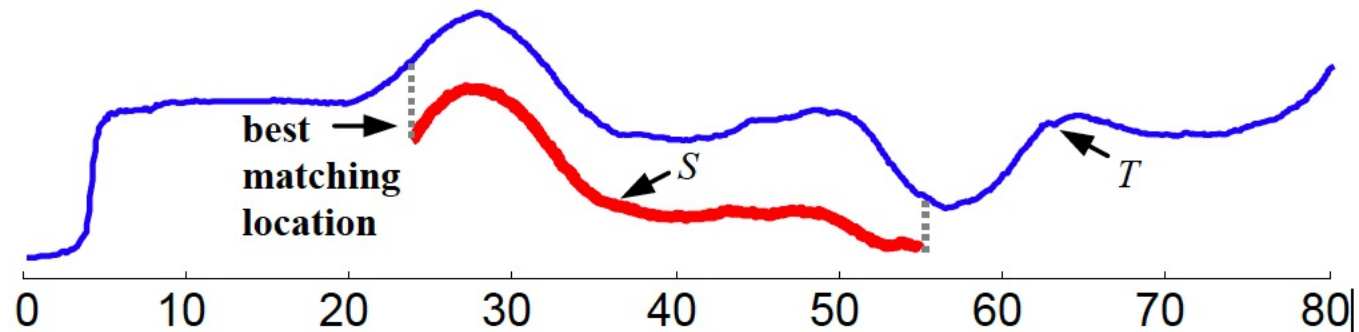


Candidates Pool



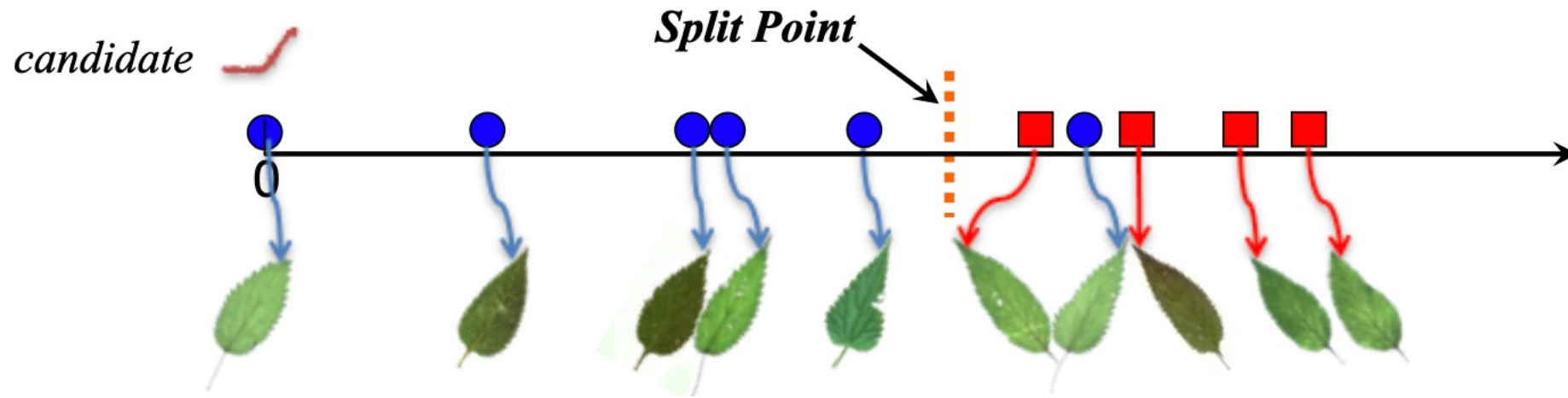
Distance with a Subsequence

- 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 nonnegative value d , which is the distance from T to S .
- $SubsequenceDist(T, S) = \min(Dist(S, S')), \text{ for } S' \in S_T^{|S|}$
- 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 .

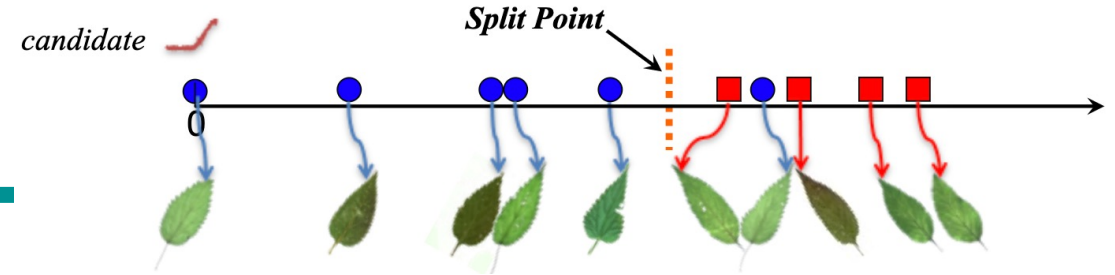


Testing The Utility of a Candidate Shapelet

- 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

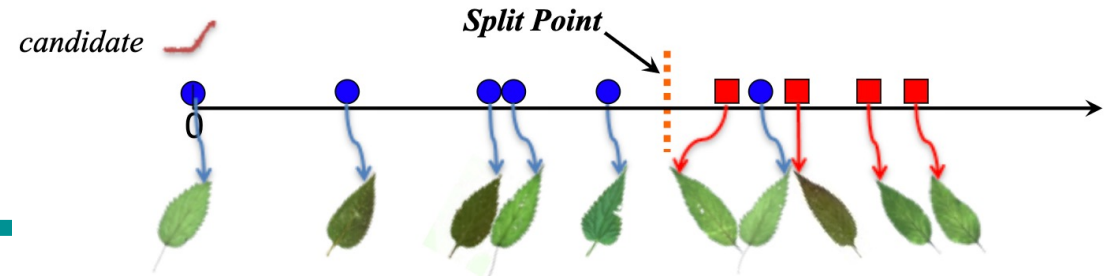


Entropy



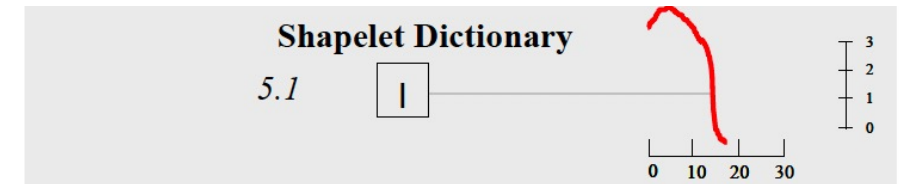
- A TS dataset D consists of two classes, A and B.
- Given that the proportion of objects in class A is $p(A)$ and the proportion of objects in class B is $p(B)$,
- The **Entropy** of D is: $I(D) = -p(A)\log(p(A)) - p(B)\log(p(B))$.
- Given a strategy that divides the D into two subsets D_1 and D_2 , 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

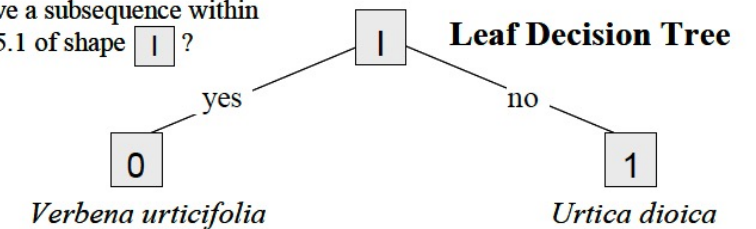


Split point
distance from
shapelet = 5.1

- 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 .



Does Q have a subsequence within a distance 5.1 of shape I ?

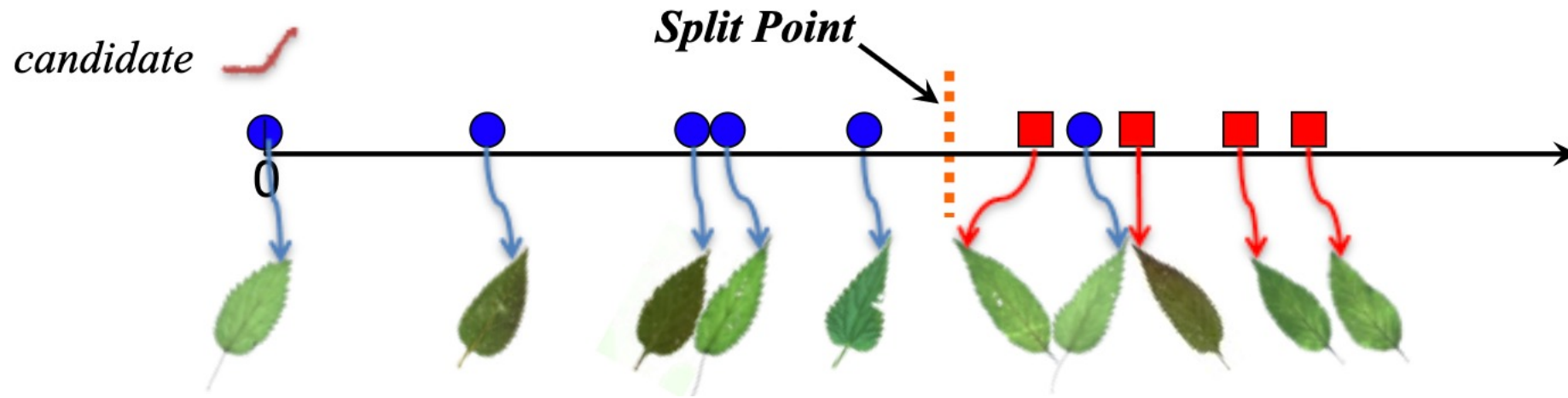


Problem

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

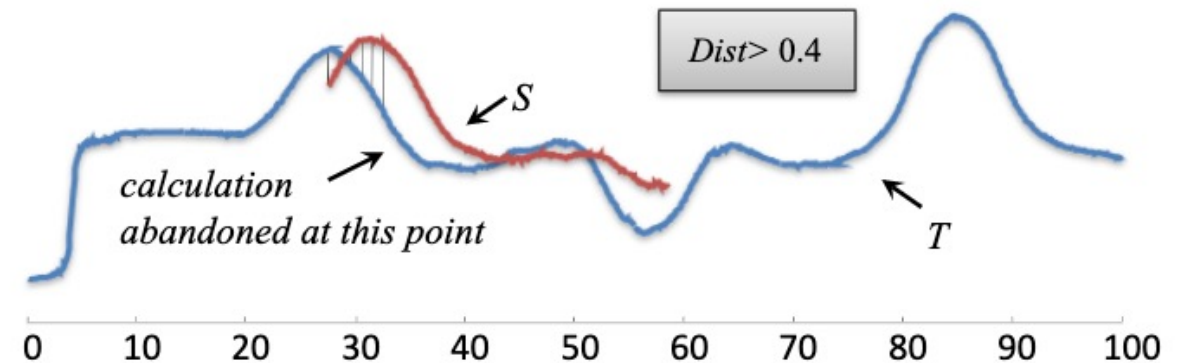
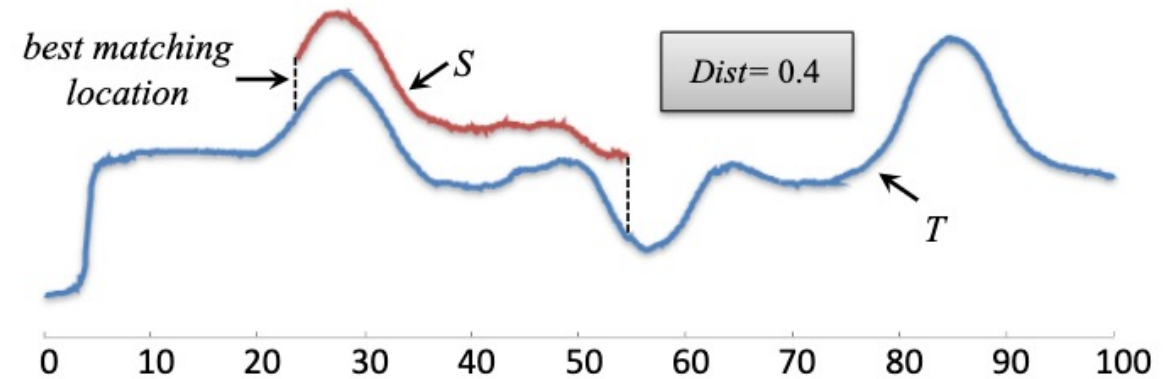
Speedup

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



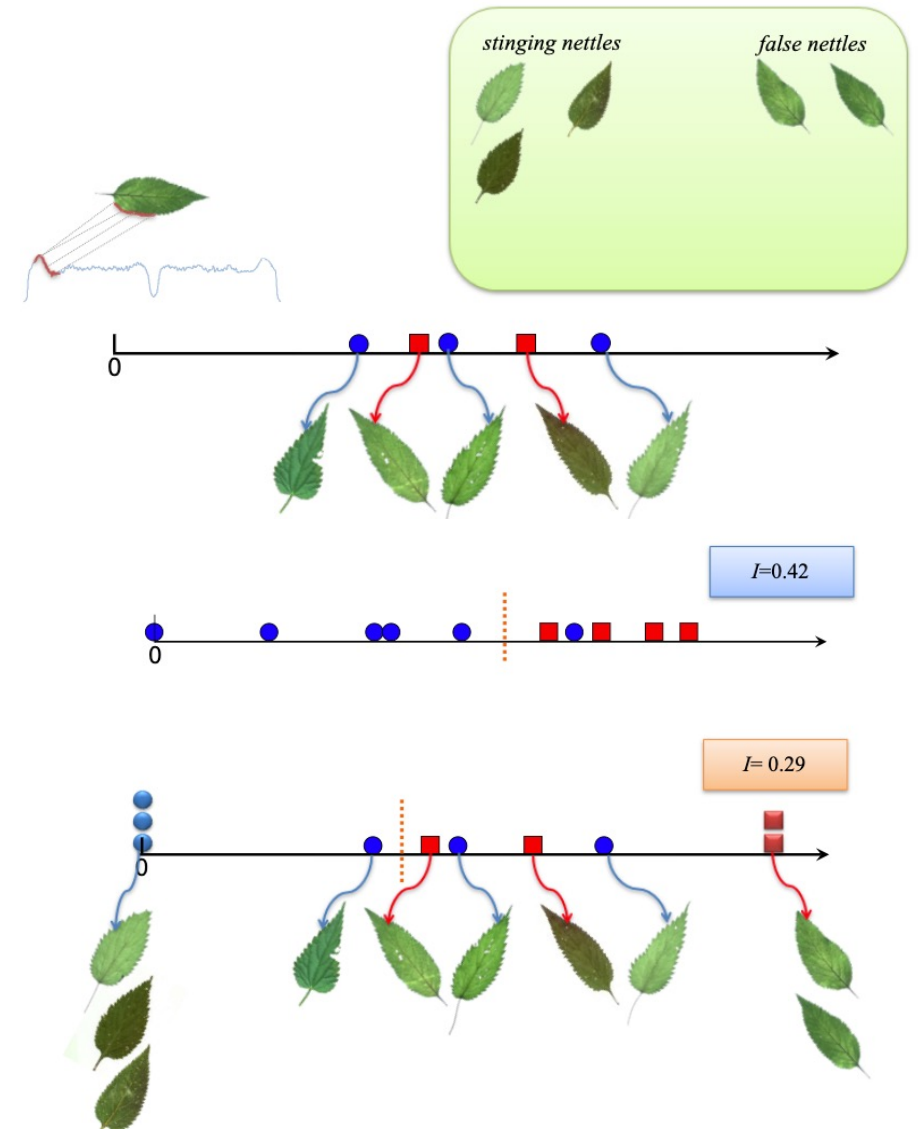
Distance Early Abandon

- We only need the minimum distance.
- Method
 - Keep the best-so-far distance
 - Abandon the calculation if the current distance is larger than best-so-far.



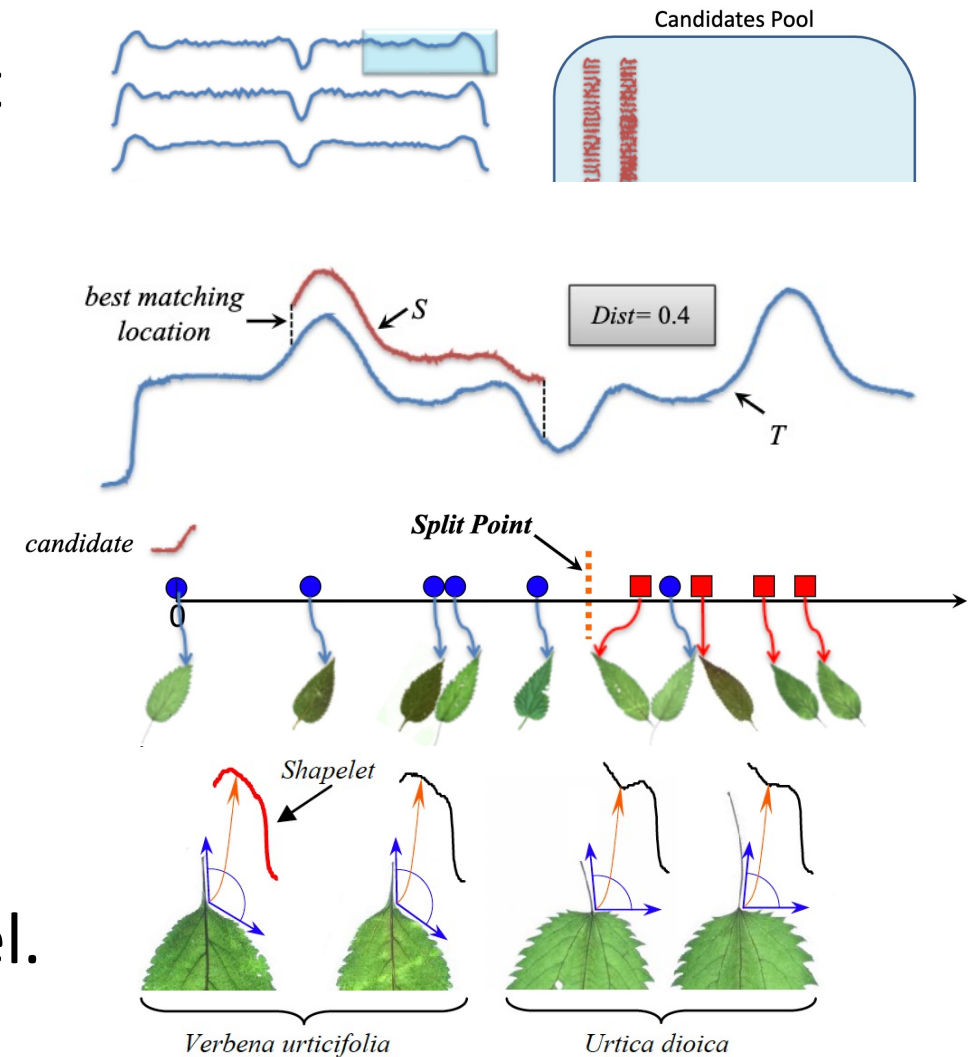
Admissible Entropy Pruning

- 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, the upper bound of information gain $<$ best candidate shapelet
 - Stop calculation
 - Try next candidate



Shapelet Summary

1. Extract all possible subsequences of a set given lengths (candidate shapelets)
2. For each candidate shapelet
 1. Calculate the distance with each time series keeping the minimum distance (best alignment)
 2. Evaluate the discriminatory effect of the shapelet through the Information Gain
3. Return the k best shapelets with the highest Information Gain.
4. Transform a dataset and train a ML model.



An Alternative Way for Extracting Shapelets

- The minimum distances (M) between Ts and Shapelets can be used as predictors to approximate the TSs label (Y) using a linear model (W):

$$\hat{Y}_i = W_0 + \sum_{k=1}^K M_{i,k} W_k, \quad \forall i \in \{1, \dots, I\}$$

- A logistic regression loss can measure the quality of the prediction:

$$\mathcal{L}(Y, \hat{Y}) = -Y \ln \sigma(\hat{Y}) - (1 - Y) \ln (1 - \sigma(\hat{Y}))$$

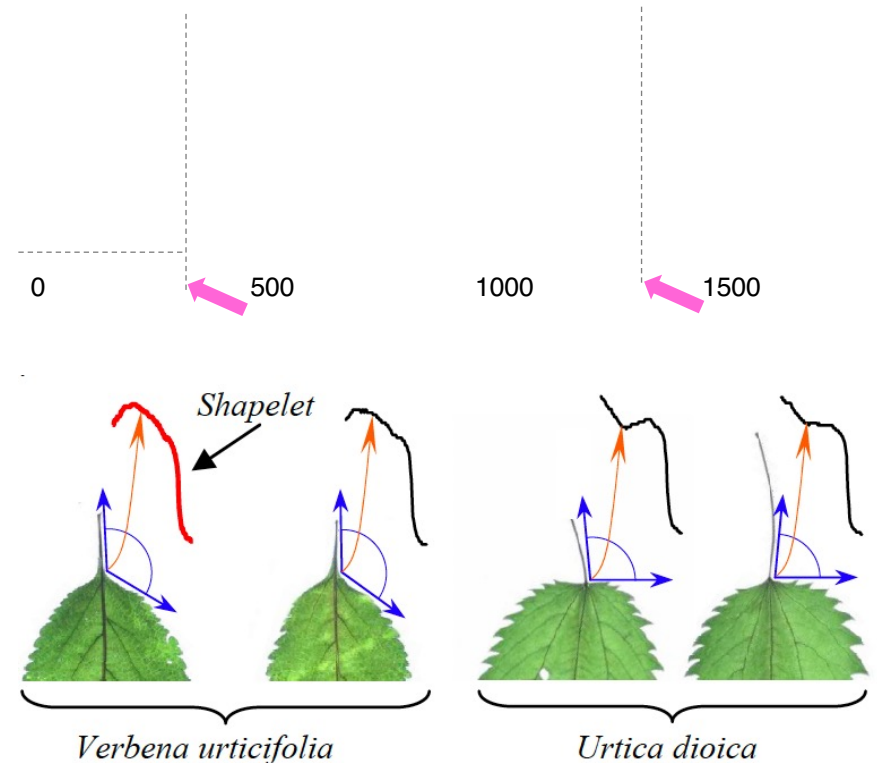
- The objective is to minimize a regularized loss function across all the instances (I) :

$$\operatorname{argmin}_{S, W} \mathcal{F}(S, W) = \operatorname{argmin}_{S, W} \sum_{i=1}^I \mathcal{L}(Y_i, \hat{Y}_i) + \lambda_W \|W\|^2$$

- We can find the optimal shapelet for the objective function in a NN fashion by updating the shapelets in the minimum direction of the objective, hence the first gradient. Similarly, the weights can be jointly updated towards minimizing the objective function.

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
- Deep learning for time series classification: a review. Hassan Ismail Fawaz et al. 2019.

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

Chin-Chia Michael Yeh, Yan Zhu, Lindmila Ulanova, Nurjahan Begam, Yifei Ding, Hoang Anh Dao, Diego Furrado Silva, Abdullh Mosen, and Eamonn Keogh
University of California, Riverside, Universidade de São Paulo, University of New Mexico
(moy003, yzhu013, hdao001, abeg001, yding007, hdao001)@ucr.edu, denqofab@unicamp.br, amosen@cs.ucr.edu

Abstract—The all-pairs-similarity-search (or similarity join) problem has been extensively studied for text and a handful of other domains. However, surprisingly little progress has been made on similarity joins for time series subsequences. The lack of progress probably stems from the difficult nature of the problem. For even modest sized datasets the obvious nested-loop algorithm can take months, but typical speed-up techniques in this domain (e.g., indexing, lower-bounding, irregularly-inequality pruning and early abandonment) do not produce one or two orders of magnitude speedup. In this work we introduce a novel scalable algorithm for time series subsequence all-pairs-similarity-search. For exceptionally large datasets, the algorithm can be trivially cast as an anytime algorithm and produce high-quality approximate solutions in reasonable time. The exact similarity join algorithm computes the answer to the time series self- and time series discord problem as a side-effect, and our algorithm incidentally provides the fastest known algorithm for both these extensively-studied problems. We demonstrate the utility of our ideas for many time series data mining problems, including motif discovery, anomaly discovery, shapelet discovery, semantic segmentation, density estimation, and contrast set mining.

Keywords—Time Series; Similarity Joins; Motif Discovery

1. INTRODUCTION

The all-pairs-similarity-search (also known as similarity join) problem comes in several variants. The basic task is: Given a collection of data objects, retrieve the nearest neighbor for each object in the text domain. The algorithm has applications in a host of problems, including community discovery, duplicate detection, collaborative filtering, clustering, and query refinement [1]. While virtually all text processing algorithms have autotuned to time series data mining, there has been surprisingly little progress on Time Series subsequence All-Pairs-Similarity-Search (TSAPSS).

We believe that this lack of progress stems not from a lack of interest in this useful primitive, but from the daunting nature of the problem. Consider the following example that reflects the needs of an industrial collaborator. A boiler's chemical refinery reports pressure once a minute. After a year, we have a time series of length 524,608. A plant manager may wish to do a similarity self-join on this data with week-long subsequences (10,800) to discover operating regimes (summer vs. winter vs. light distilling vs. heavy distilling, etc). The obvious nested loop algorithm requires 112,800,602,560 Euclidean distance computations. If we assume each one takes 0.0001 seconds, then the join will take 11.83 days. The core combination of this work is to show that we can reduce this time to 8-3 hours, using an off-the-shelf desktop computer. Moreover, we show that this join can be computed and/or updated incrementally. This work could maintain this join execution forever on a standard

desktop, even if the data arrival frequency was much faster than one a minute.

Our algorithm uses an ultra-fast similarity-search algorithm under a novel notion of Euclidean distance as a subsequence, exploiting the overlap between subsequences using the classic Fast Fourier Transform (FFT) algorithm.

Our method has the following advantages/features:

- It is exact, providing no false positives or false dismissals.
- It is simple and parameter-free. In contrast, the more general metric space APSS algorithms require building and tuning spatial access methods and/or hash functions.
- Our algorithm requires an inconsequential space overhead, just $O(n)$ with a small constant factor.
- While our exact algorithm is extremely scalable, for extremely large datasets we can compute the results in an anytime fashion, allowing ultra-fast approximate solutions.
- Having computed the similarity join for a dataset, we can incrementally update it very efficiently. In many domains this means we can effectively maintain exact joins on streaming data forever.
- Our method provides full joins, eliminating the need to specify a similarity threshold, which as we will show, is a near impossible task in this domain.
- Our algorithm is embarrassingly parallelizable, both on multicore processors and on distributed systems.

Time Series Shapelets: A New Primitive for Data Mining

Lexiang Ye
Dept. of Computer Science & Engineering
University of California, Riverside, CA 92521
leyang@cs.ucr.edu

Eamonn Keogh
Dept. of Computer Science & Engineering
University of California, Riverside, CA 92521
eamonn@cs.ucr.edu

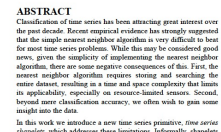


Figure 1: Sample of leaves from two species. Note that several leaves have the insect-bite damage.

Suppose we wish to build a classifier to distinguish these two plants, what features should we use? Since the area-variability of color and size within each class completely dwarfs the inter-variability between classes, our best hope is based on the shape of the leaves. However, as we can see in Figure 1, the difference in the global shape are very subtle. Furthermore, it is very common for leaves to have distortions or “irregularities” due to insect damage, and these are likely to confuse any global measures of shape. Instead we attempt the following. We first convert each leaf into a one-dimensional representation as shown in Figure 2.



Figure 2: A shape can be converted into a one-dimensional “time series” representation. The reason for the highlighted section of the time series will be made apparent shortly.

Such representations have been successfully used for the classification, clustering and outlier detection of shapes in recent years [2]. However, here we find that using a nearest neighbor classifier with either the (position-normalized) Euclidean distance or Dynamic Time Warping (DTW) distance does not significantly outperform random guessing. The reason for the poor performance of these otherwise very competitive classifiers seems to be due to the fact that the data is somewhat noisy (i.e. insect holes, and different insect lengths), and this noise is enough to swamp the subtle differences in the shapes.

arXiv:1809.04356v4 [cs.LG] 14 May 2019

This is the author's version of an article published in Data Mining and authenticated version is available online at: https://doi.org/10.1007/978-1-4939-9809-6_55

Deep learning for time series classification

Hassan Ismail Fawaz¹ · Germain Forestier^{1,2} · Jonathan W. Hussane Idoumghar¹ · Pierre-Alain Muller¹

Abstract Time Series Classification (TSC) is an important and challenging task. With the increase of time series data availability, hundreds of TSC algorithms have been proposed since 2015 (Bagnall et al., 2017). Due to their natural temporal ordering, time series data are present in almost every task that requires some sort of human cognitive process (Långkvist et al., 2014). In fact, any classification problem, using data that is registered taking into account some notion of ordering, can be cast as a TSC problem (Cristian Borges Gombos, 2017). Time series are encountered in many real-world applications ranging from electronic health records (Rajkumar et al., 2018) and human activity recognition (Nweke et al., 2018; Wang et al., 2018) to acoustic scene classification (Nwe et al., 2017) and cyber-security (Susto et al., 2018). In addition, the diversity of the datasets’ types in the UCR/UEA archive (Chen et al., 2015b; Bagnall et al., 2017) (the largest repository of time series datasets) shows the different applications of the TSC problem.

Keywords Deep learning · Time series · Classification · Review

1 Introduction

During the last two decades, Time Series Classification (TSC) has been considered as one of the most challenging problems in data mining (Yang and Wu, 2006; Ealing and Agon, 2012). With the increase of temporal data availability (Silva et al., 2018), hundreds of TSC algorithms have been proposed since 2015 (Bagnall et al., 2017). Due to their natural temporal ordering, time series data are present in almost every task that requires some sort of human cognitive process (Långkvist et al., 2014). In fact, any classification problem, using data that is registered taking into account some notion of ordering, can be cast as a TSC problem (Cristian Borges Gombos, 2017). Time series are encountered in many real-world applications ranging from electronic health records (Rajkumar et al., 2018) and human activity recognition (Nweke et al., 2018; Wang et al., 2018) to acoustic scene classification (Nwe et al., 2017) and cyber-security (Susto et al., 2018). In addition, the diversity of the datasets’ types in the UCR/UEA archive (Chen et al., 2015b; Bagnall et al., 2017) (the largest repository of time series datasets) shows the different applications of the TSC problem.

✉ H. Ismail Fawaz
E-mail: hassan.ismail.fawaz@uniba.fr
¹IRIMAS, Université Haute Alsace, Mulhouse, France
²Faculty of IT, Monash University, Melbourne, Australia

References

- Selective review of offline change point detection methods. Truong, C., Oudre, L., & Vayatis, N. (2020). *Signal Processing*, 167, 107299.
- Time Series Analysis and Its Applications. Robert H. Shumway and David S. Stoffer. 4th edition. (<https://www.stat.pitt.edu/stoffer/tsa4/tsa4.pdf>)
- Mining Time Series Data. Chotirat Ann Ratanamahatana et al. 2010. (https://www.researchgate.net/publication/227001229_Mining_Time_Series_Data)
- Dynamic Programming Algorithm Optimization for Spoken Word Recognition. Hiroaki Sakode et al. 1978.
- Experiencing SAX: a Novel Symbolic Representation of Time Series. Jessica Line et al. 2009
- Compression-based data mining of sequential data. Eamonn Keogh et al. 2007.

