Time Series - Shapelet

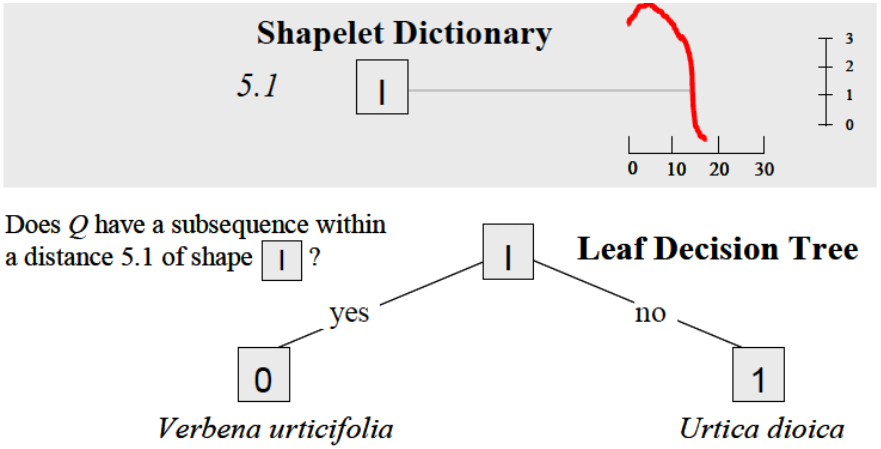
Time Series Classification

- 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.
- Does the class depend on the overall behavior of the series, or can a **short local segment be enough**?

Shapelet-based Classification

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

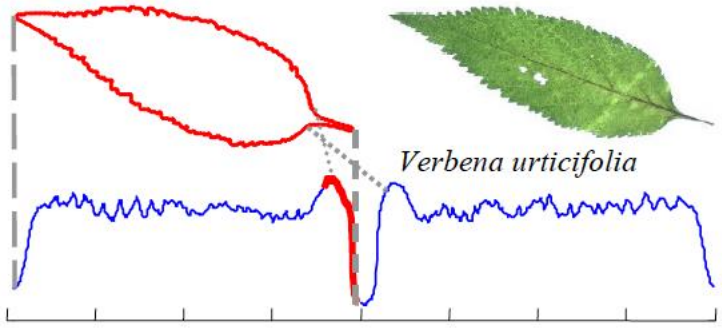
2. Use it as input for machine learning classifiers.

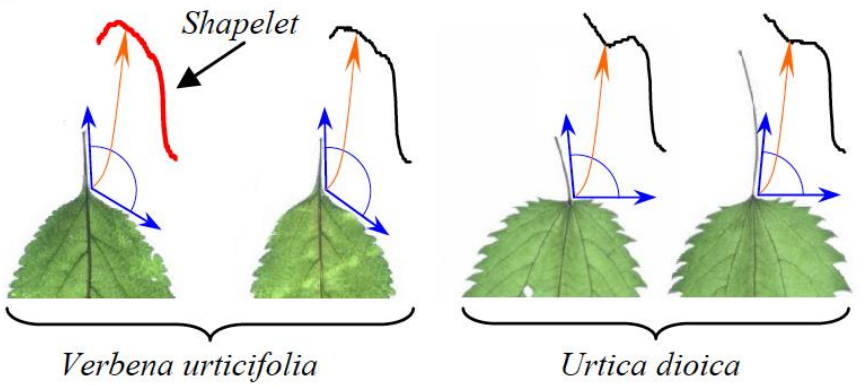




3.2	8.7
1.4	7.9
6.7	4.2
9.2	3.4





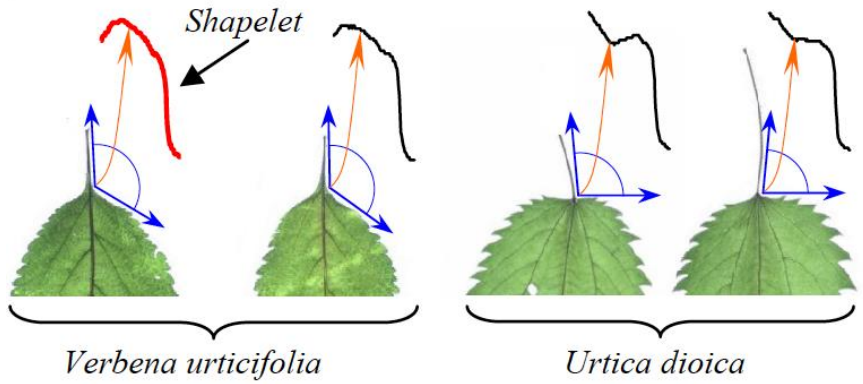


Time Series Shapelets

- 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

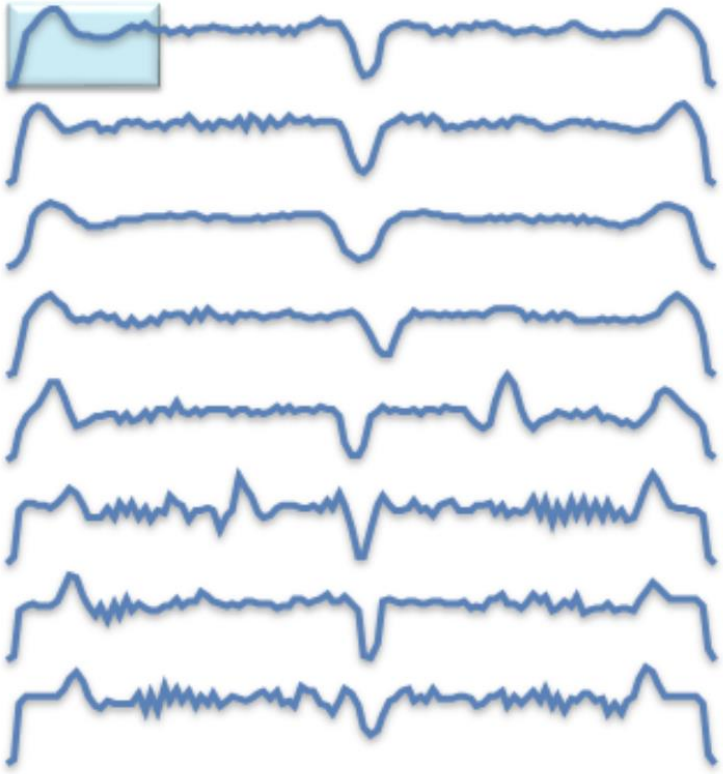
```
FindingShapeletBF (dataset D, MAXLEN, MINLEN)
1    candidates ← GenerateCandidates(D, MAXLEN, MINLEN)
2    bsf_gain ← 0
3    For each S in candidates
4        gain ← CheckCandidate(D, S)
5        If gain > bsf_gain
6            bsf_gain ← gain
7            bsf_shapelet ← S
8        EndIf
9    EndFor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

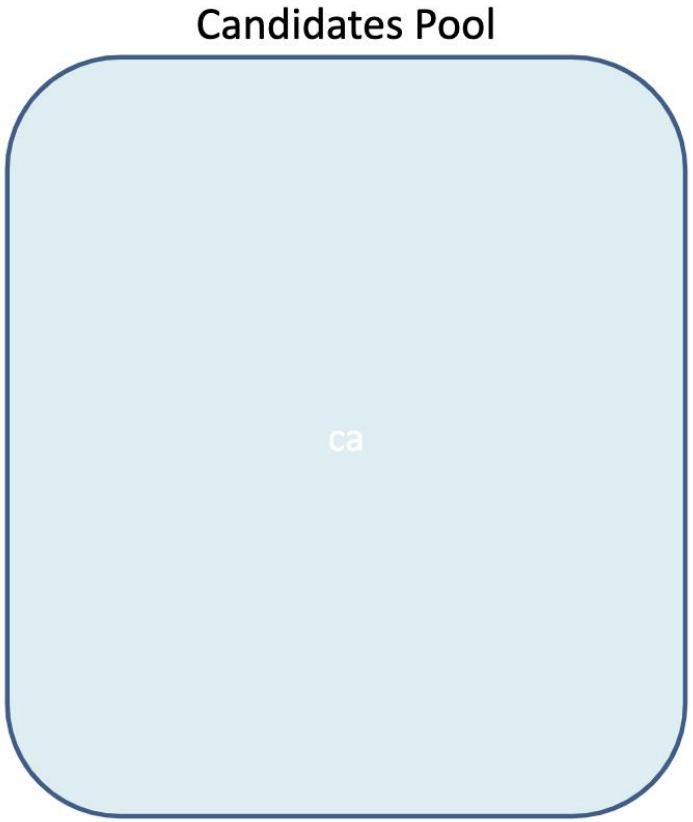
Generate Candidate

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

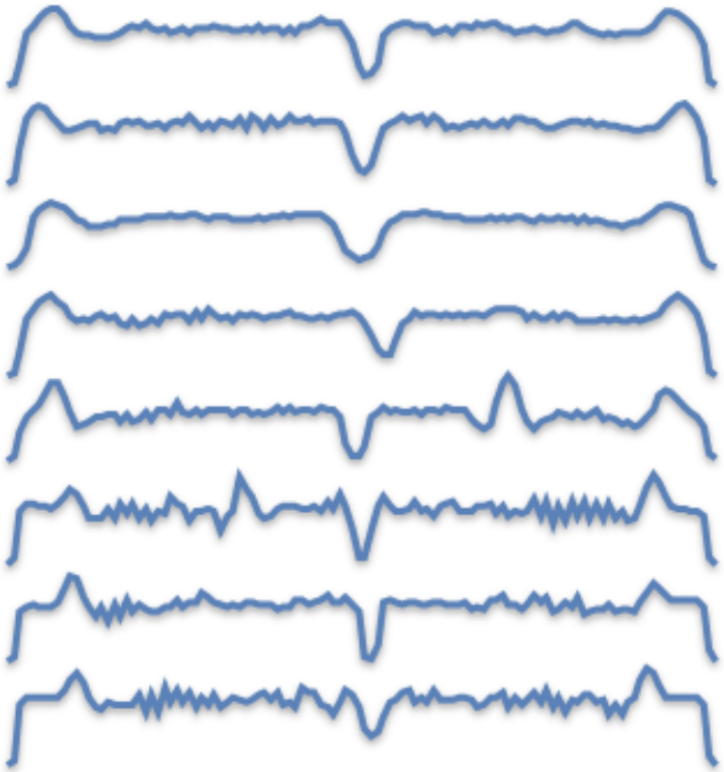
GenerateCandidates (dataset **D**, *MAXLEN*, *MINLEN*)	
1	*pool* ← Ø
2	*l* ← *MAXLEN*
3	**While** *l* ≥ *MINLEN*
4	**For** *T* **in D**
5	*pool* ← *pool* ∪ $\mathbf{S}_T^l$
6	**EndFor**
7	*l* ← *l* - 1
8	**EndWhile**
9	**Return** *pool*

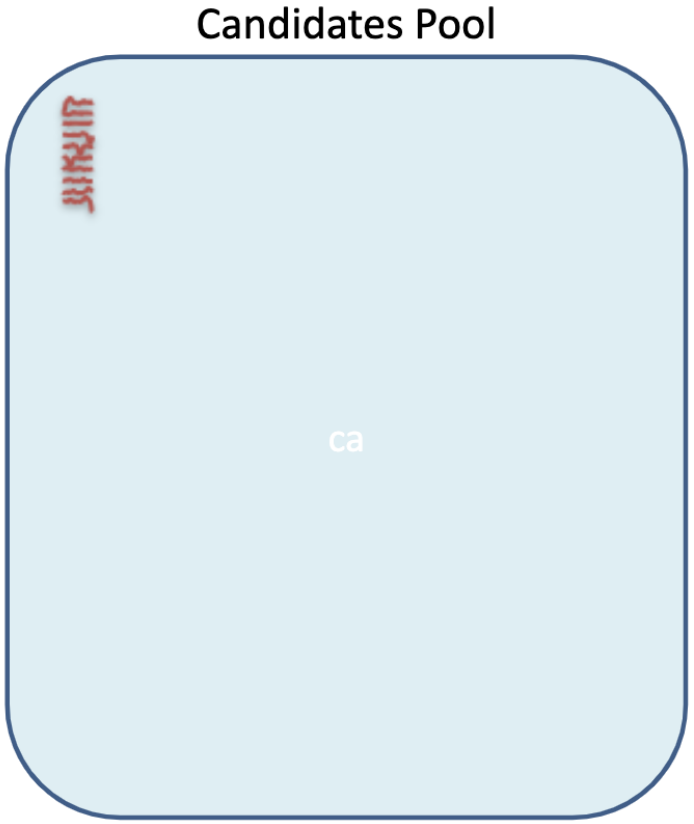
Extract Subsequences of all Possible Lengths



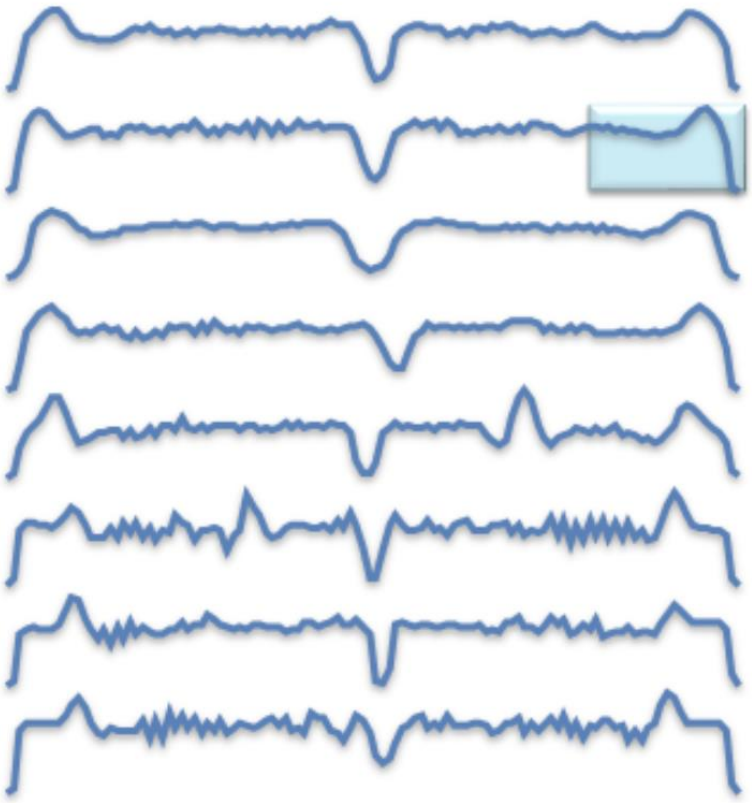


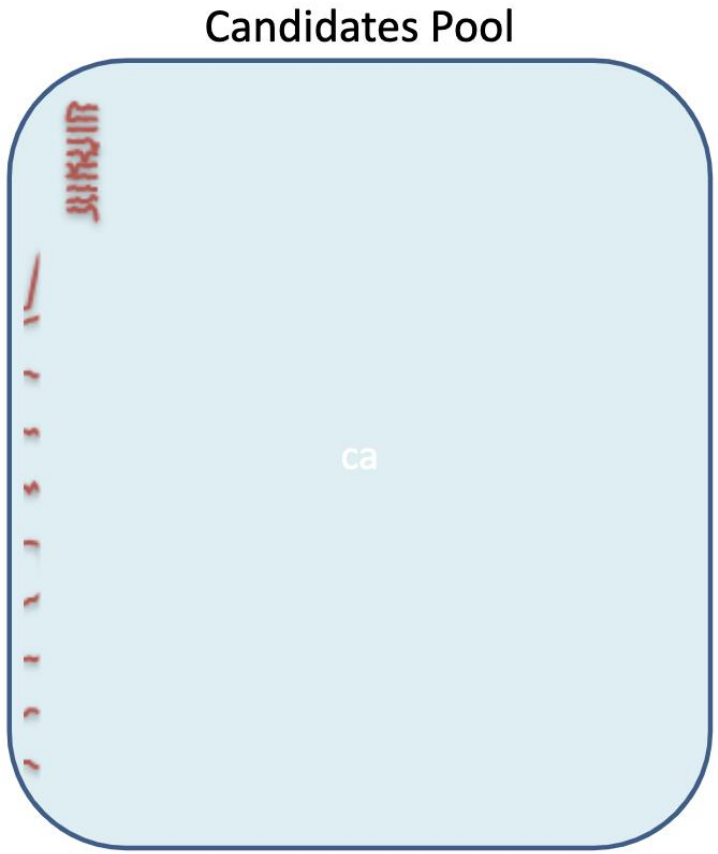
Extract Subsequences of all Possible Lengths



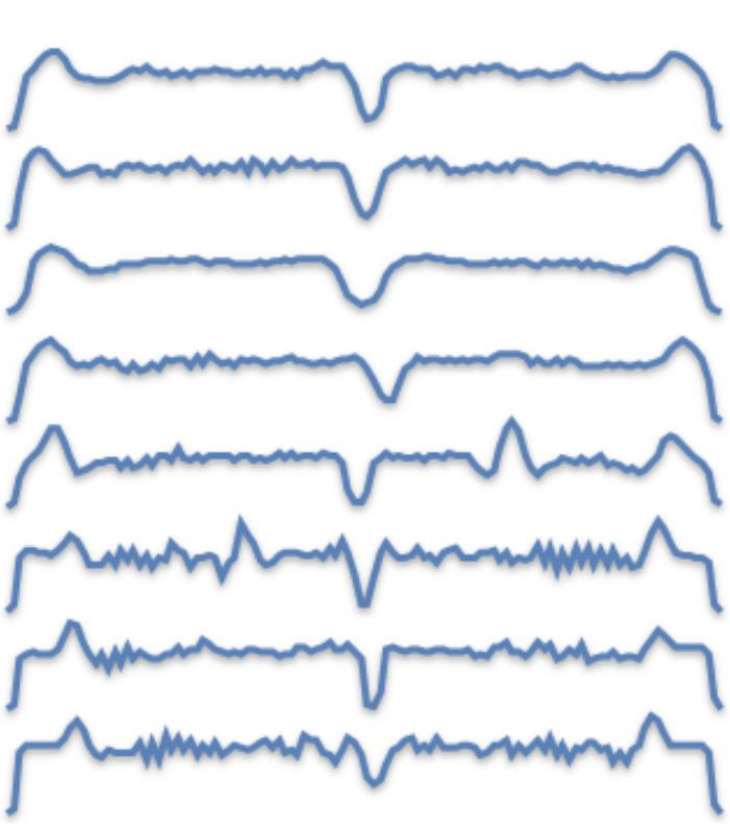


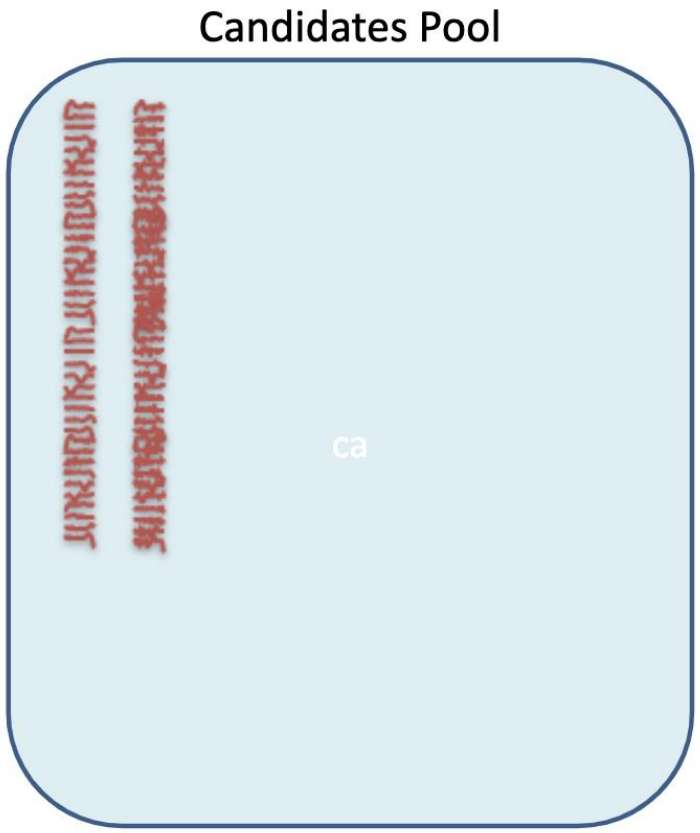
Extract Subsequences of all Possible Lengths



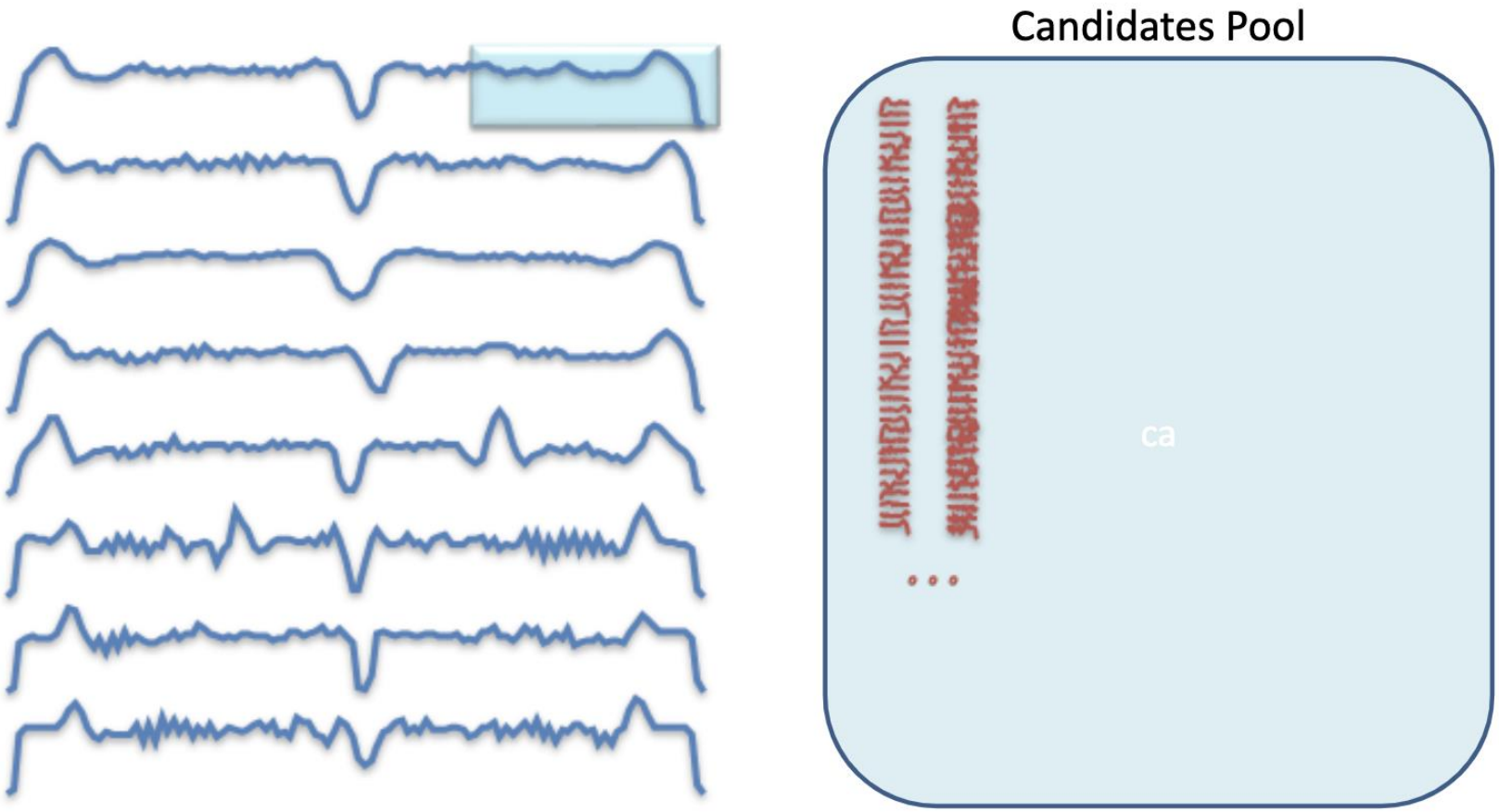


Extract Subsequences of all Possible Lengths





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*)

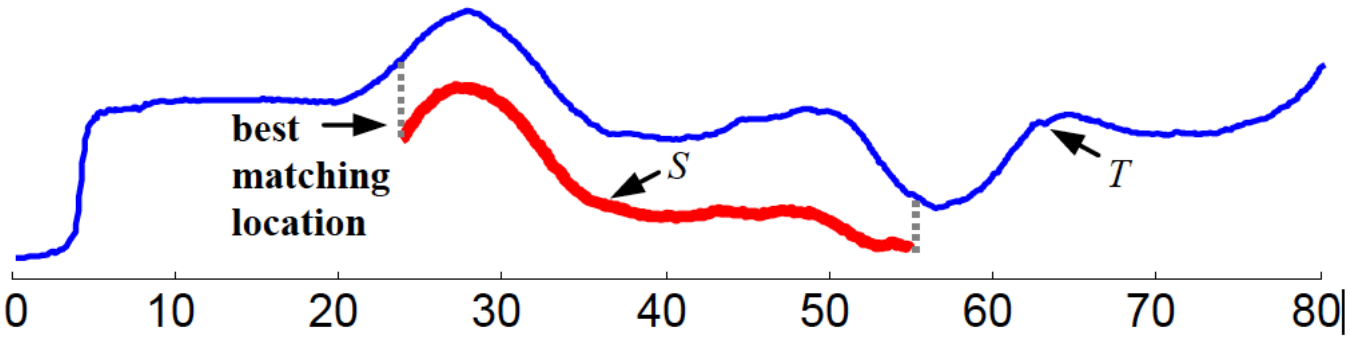
- Inserts all of the time series objects into the histogram objects_histogram according to the distance from the time series object to the candidate
- Calculate Information Gain

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.

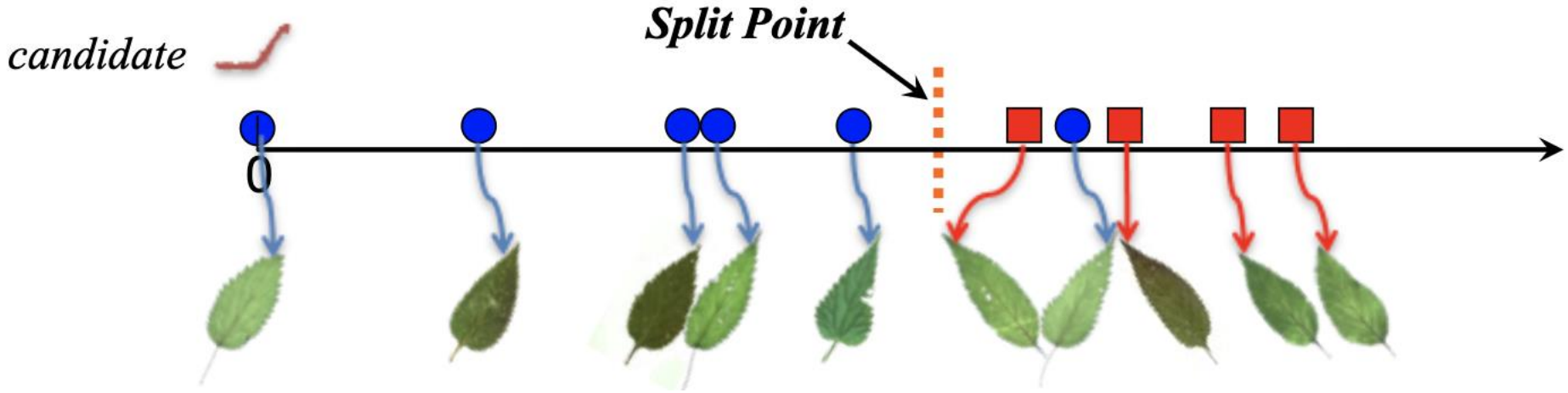


Check Candidates with IG

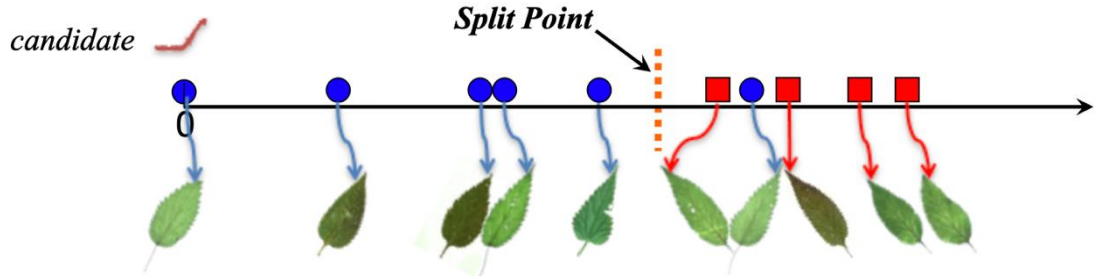
CalculateInformationGain (distance histogram *obj_hist*)	
1	*split_dist* ← OptimalSplitPoint(*obj_hist*)
2	$\mathbf{D_1} \leftarrow \emptyset$, $\mathbf{D_2} \leftarrow \emptyset$
3	**For** *d* **in** *obj_hist*
4	**If** *d.dist* < *split_dist*
5	$\mathbf{D_1} \leftarrow \mathbf{D_1} \cup d.objects$
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

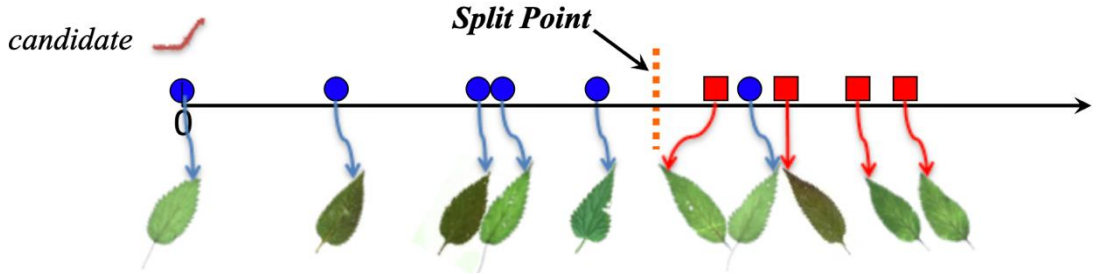


Entropy



- 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

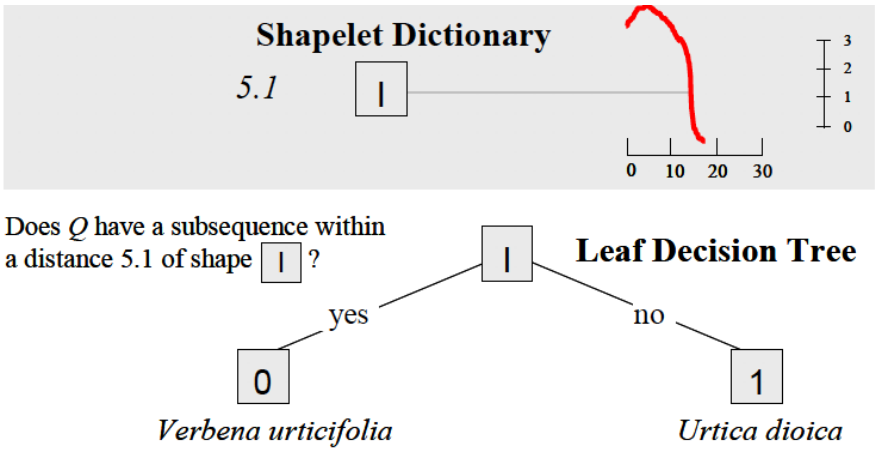


Split point distance from shapelet = 5.1

- 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

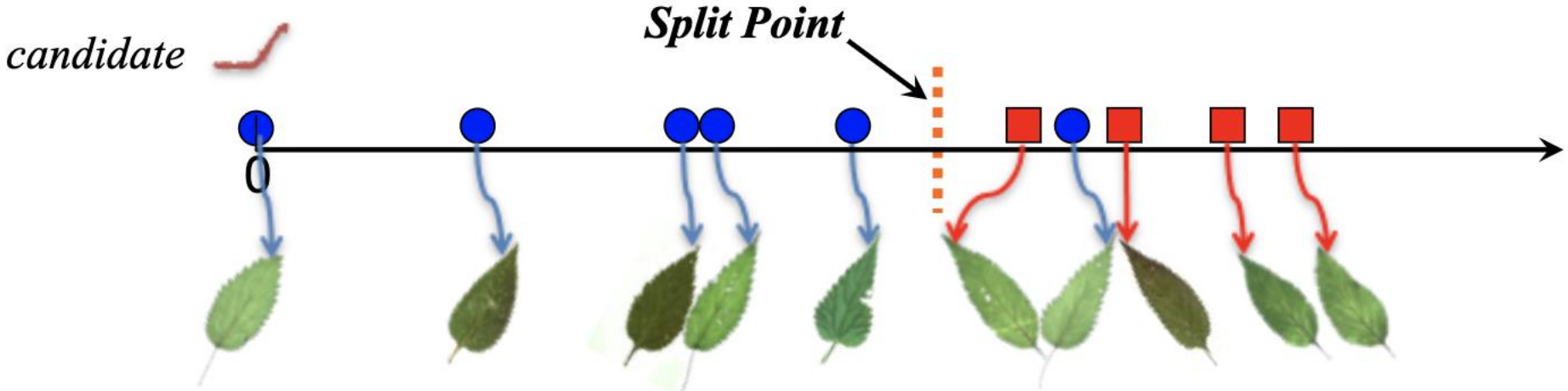
- The total number of candidate is $\sum_{l=MINLEN}^{MAXLEN} \sum_{T_i \in D} (|T_i| - l + 1)$

- 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

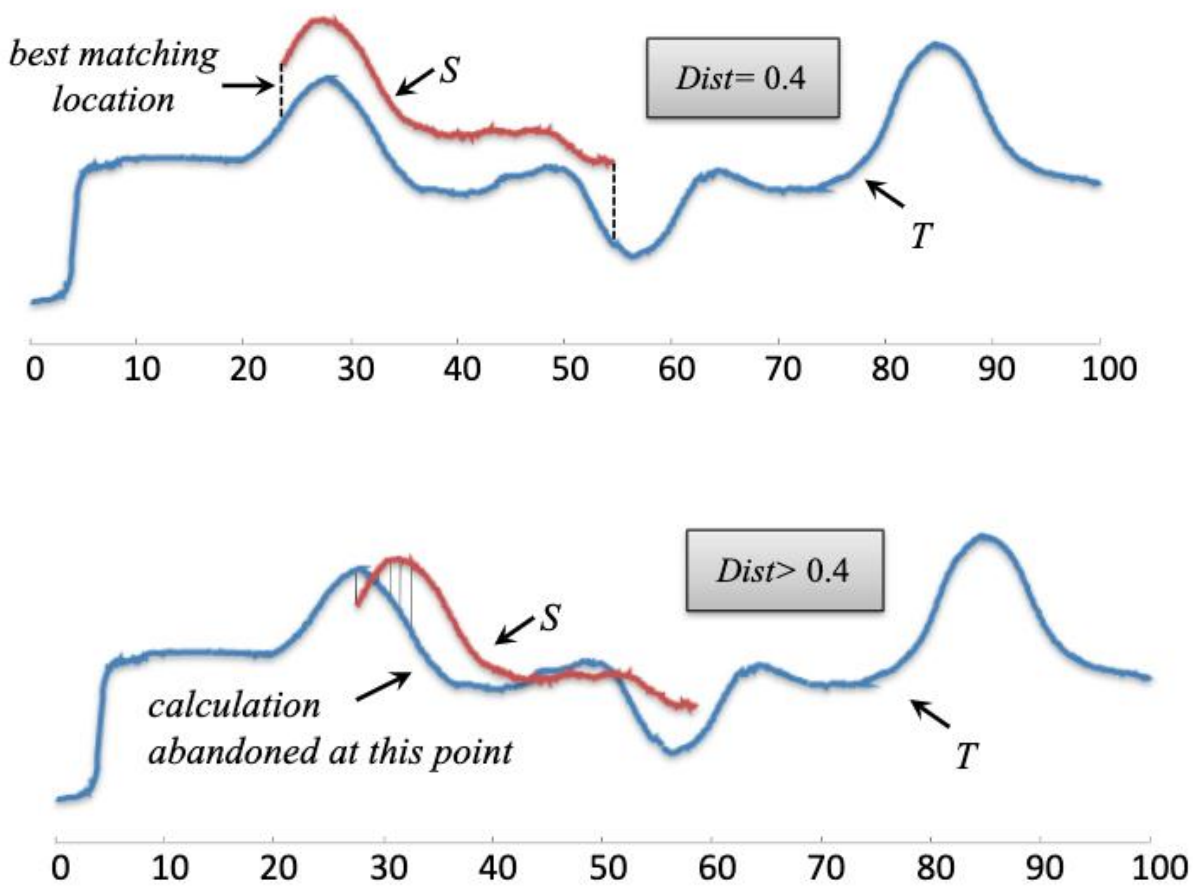
Speedup

- 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



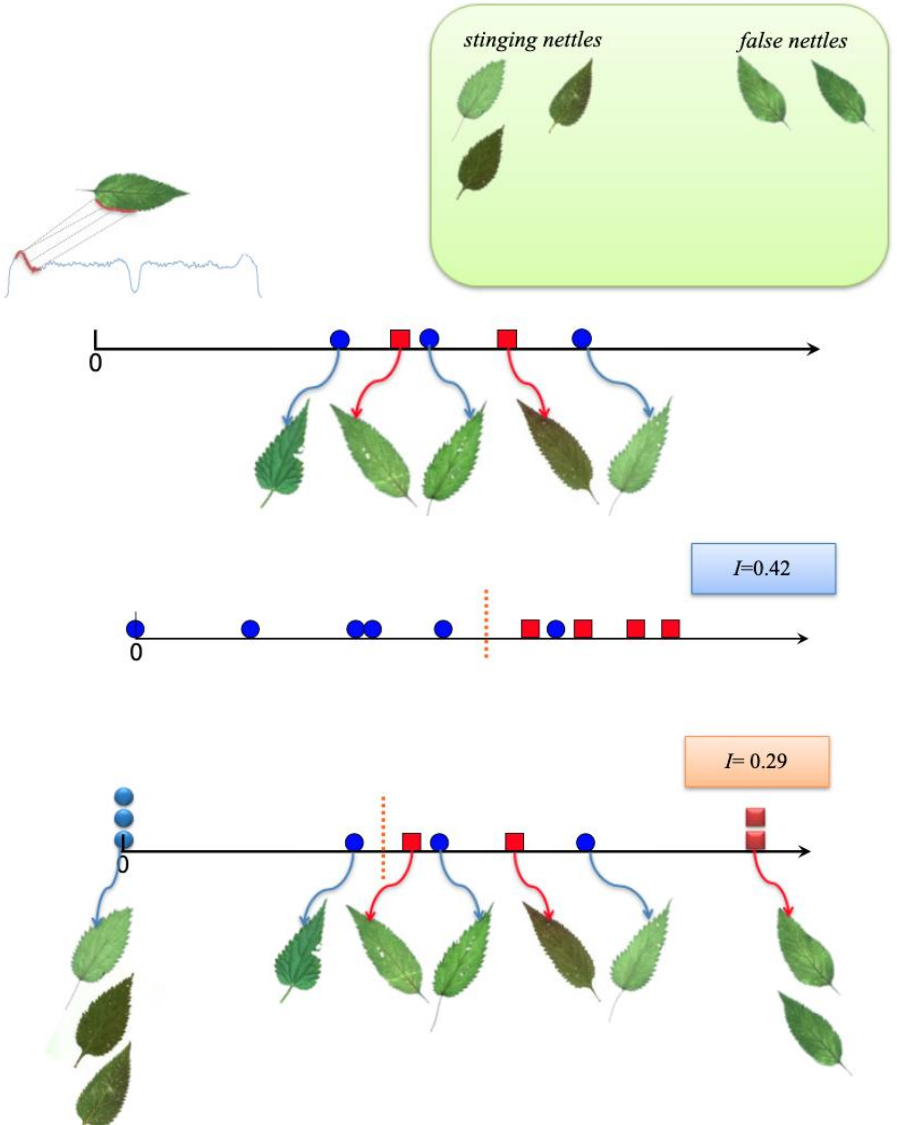
Distance Early Abandon

- 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



References

Time Series Shapelets: A New Primitive for Data Mining. Lexiang Ye and Eamonn Keogh. 2016.



Time Series Shapelets: A New Primitive for Data Mining

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

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

ABSTRACT

Classification of time series has been attracting great interest over the past decade. Recent empirical evidence has strongly suggested that the simple nearest neighbor algorithm is very difficult to beat for most time series problems. While this may be considered good news, given the simplicity of implementing the nearest neighbor algorithm, there are some negative consequences of this. First, the nearest neighbor algorithm requires storing and searching the entire dataset, resulting in a time and space complexity that limits its applicability, especially on resource-limited sensors. Second, beyond mere classification accuracy, we often wish to gain some insight into the data.

In this work we introduce a new time series primitive, *time series shapelets*, which addresses these limitations. Informally, shapelets are time series subsequences which are in some sense maximally representative of a class. As we shall show with extensive empirical evaluations in diverse domains, algorithms based on the time series shapelet primitives can be interpretable, more accurate and significantly faster than state-of-the-art classifiers.

Categories and Subject Descriptors

H.2.8 [Database Management]: Database Applications – Data Mining

General Terms

Algorithms, Experimentation

1. INTRODUCTION

While the last decade has seen a huge interest in time series classification, to date the most accurate and robust method is the simple nearest neighbor algorithm [4][12][14]. While the nearest neighbor algorithm has the advantages of simplicity and not requiring extensive parameter tuning, it does have several important disadvantages. Chief among these are its space and time requirements, and the fact that it does not tell us anything about *why* a particular object was assigned to a particular class.

In this work we present a novel time series data mining primitive called *time series shapelets*. Informally, shapelets are time series subsequences which are in some sense maximally representative of a class. While we believe shapelets can have many uses in data mining, one obvious implication of them is to mitigate the two weaknesses of the nearest neighbor algorithm noted above.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
KDD'09, June 28–July 1, 2009, Paris, France
Copyright 2009 ACM 978-1-60558-495-9/09/06…$5.00.

Because we are defining and solving a new problem, we will take some time to consider a detailed motivating example. Figure 1 shows some examples of leaves from two classes, *Urtica dioica* (stinging nettles) and *Verbena urticifolia*. These two plants are commonly confused, hence the colloquial name "false nettle" for *Verbena urticifolia*.

Figure 1: Samples 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 intra-variability of color and size within each class completely dwarfs the inter-variability between classes, our best hope is based on the shapes of the leaves. However, as we can see in Figure 1, the differences in the global shape are very subtle. Furthermore, it is very common for leaves to have distortions or "occlusions" 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 [8]. However, here we find that using a nearest neighbor classifier with either the (rotation invariant) 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 bites, and different stem lengths), and this noise is enough to swamp the subtle differences in the shapes.