Efficient class-specific shapelets learning for interpretable time series classification

Data mining presentation P17

Dalla Noce Niko, Lombardi Giuseppe, Ristori Alessandro

17 dicembre 2021

Main sections

1) Introduction

Quick overview of the current state of the art methods for shapelet-based time series classification

2) Shapelet candidate method

Indroducing the new approach for selecting the initial shapelets for the learning phase.

3) Learning

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh.

4) Results

The results obtained by the model compared with other classifiers.

Introduction

Time series classification is a critical problem in data mining, with many applications in diverse areas, including medical, biological, financial, engineering, and industrial.

Shapelet-based approaches have attracted increasing attention in recent years due to the **comprehensive interpretability** of the classification results. This is fine, but how to **discover the shapelets**?

- By **checking all possible subsequences** with $O(M^2N^4)$ time, where M is the number of time series in the training dataset and N is the length of the training series, or

- **directly learn the true shapelets** rather than searching from all the problem space, which can achieve competitive classification accuracy by **finding the near-to-optimal shapelets** which do not exist in the raw time series.
  The learning-based approach requires $O(IN_uMN^2)$ time where I is the number of iterations, N_u is the number of shapelets to be updated per iteration.

The study

The main contributions of this study are summarized as follows:

- **learn class-specific shapelets** rather than **shared shapelets** in order to reduce the number of calculations of shapelet updating;

- propose a class-specific **shapelet candidates discovery** method to **automatically determine** the **number**, **length**, and **initial values** of the **shapelets** in the **learning model**. It can **dramatically reduce** the number of **calculations** of **shapelet updating** N_u by reducing the number of shapelets K, and effectively decrease the number of iterations I.

Some definitions

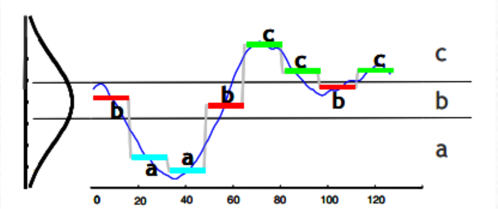
- *Distance between two time series:*
  Let T_m and T_n be two time sequences of length L_m and L_n where $L_m \geq L_n$, we denote the point-wise distance between a subsequence of T_m which starts at p and has length L_n, i.e., T_m^{p,L_n} , and the whole series T_n as $Dis^p(T_m, T_n)$, i.e., $Dis^p(T_m, T_n) = \frac{1}{L_n}\sum_{l=1}^{L_n}\left(T_m^{p+l-1} - T_n^l\right)^2$, where $1 \leq p \leq L_m - L_n + 1$.
  Then, we **define the distance between T_m and T_n** as the minimum point-wise distance between the L_n-length subsequence of T_m and T_n, i.e., $Dis(T_m, T_n) = min_{p=1,2,\ldots,J} \frac{1}{L_n} Dis^p(T_m, T_n)$, where $J = L_m - L_n + 1$.
- *Pattern:*
  We define a **time series pattern P** as a series that can **reflect certain features** of the time series.
  Note that the pattern could be, but is **not restricted** to the **subsequence** of the **time series**.

- *Class-Specific Shapelet(CSS)):*
  The **pattern** that can **maximally reflect** the **distinguishing feature** of a **specific category** is called the class-specific shapelet of that class. A **CSS** has a **great discriminatory power** to determine **whether** or not a **time series belongs** to a **category**.

Approximate motif discovery

The approximate motif discovery technique is helpful to find the **shapelet candidates.**
It works as follow:

1) the **input time series** is **discretized** to a word sequence by Symbolic Aggregate Approximation (**SAX**).



2) Next, a grammar induction algorithm, called **Sequitur**, is conducted on the transformed sequence to **extract repeated series** of **words**. **Sequitur** is a **string compression** algorithm that recursively **replaces repeated substrings with grammatical rules** in **linear time** and **space**.
Each **expanded rule** becomes a **repeated word subsequence** of the input.

An example of sequitur.

Grammar rule	Expanded rule
$R_0 \rightarrow R_1\ cda\ R_1\ R_2$	$\underbrace{\underbrace{abb\ abc}_{R_2}\ abd}_{R_1}\ cda\ \underbrace{\underbrace{abb\ abc}_{R_2}\ abd}_{R_1}\ \underbrace{abb\ abc}_{R_2}$
$R_1 \rightarrow R_2\ abd$	$\underbrace{abb\ abc}_{R_2}\ abd$
$R_2 \rightarrow abb\ abc$	$abb\ abc$

3) At the aend, those time series **subsequences** are **discovered approximate motifs**.

Pattern selection

- The **motifs discovered contains** many **similar patterns** due to the feature of SAX and the Sequitur algorithm. These **similar patterns** have almost **no difference** in **discriminatory power**. Thus, we only **select one representative** from each **group** of **similar motifs** as a **class-specific shapelet.**

Evaluation setup

The method was **evaluated on 25 datasets of the UCR archive** commonly used by shapelet-based methods.

The **evaluation was splitted in two parts**:

- Evaluation of the **acceleration of the CSSL model** over the existing shared shapelets learning approach, Learning Shapelet (LS).
- Evaluation of the **classification performance** of the CSSL approach:
    - First **the method was compared against 18 representative classifiers** in terms of classification accuracy.
    - Then **the method was compared against state-of-the-art classifiers** designed for time-series classification problems by extracting representative timeseries features.

Efficiency evaluation (running time)

The **running time of the shapelet candidates discovery is negligible compared to the optimization process,** hence the efficiency of CSSL is mainly **determined by the second phase:**

- SAX and Sequitur are both linear algorithms.
- The clustering algorithm **processes a few small pieces of strings or sequences**.
- During the second phase, **the shapelets need to be optimized for every training time series** over hundreds of iterations.

The non-parallel algorithm is up to 70 times faster, and the parallel version is up to 622 times faster.

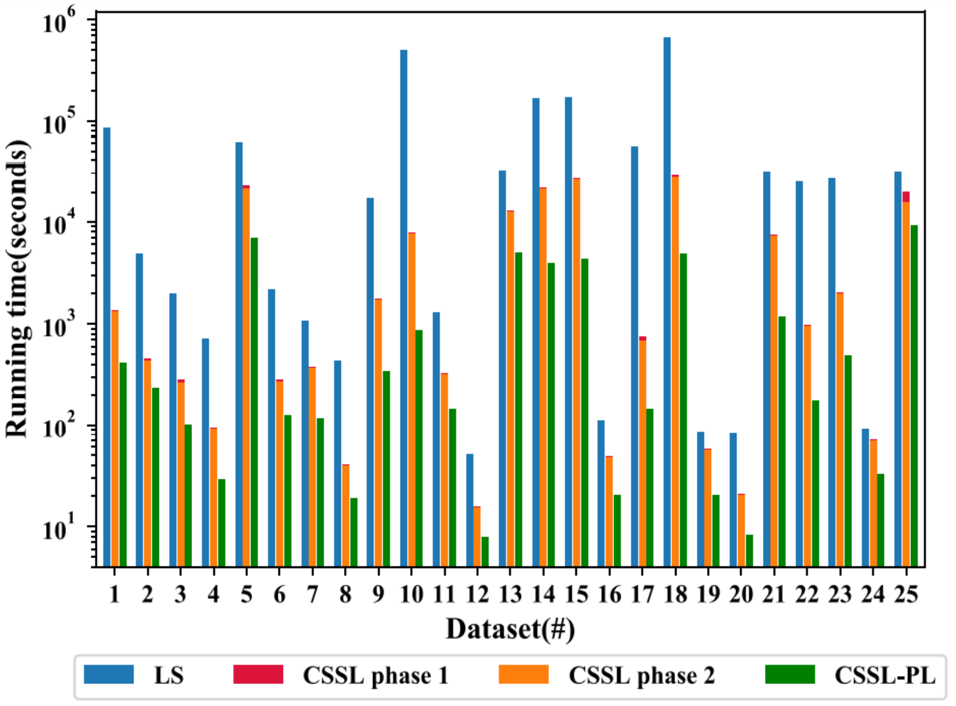


Fig. 5. The total training time of LS, CSSL, and CSSL-PL.

Efficiency evaluation (shapelets)

CSSL discovers discriminatory shapelet candidates for shapelet initialization of the classification model. These candidates are **more likely to be near the optimums** thus causing fewer iterations.

The algorithm ends up with only 500 iterations on half of the datasets, while LS requires more than 5,000 iterations (fixed learning rate) and 1,000 iterations (variable learning rate). **Fewer iterations contribute to a faster training as well**.

CSSL conducts much less shapelet updating at each iteration benefiting from the idea of learning class-specific and as few as possible shapelets.

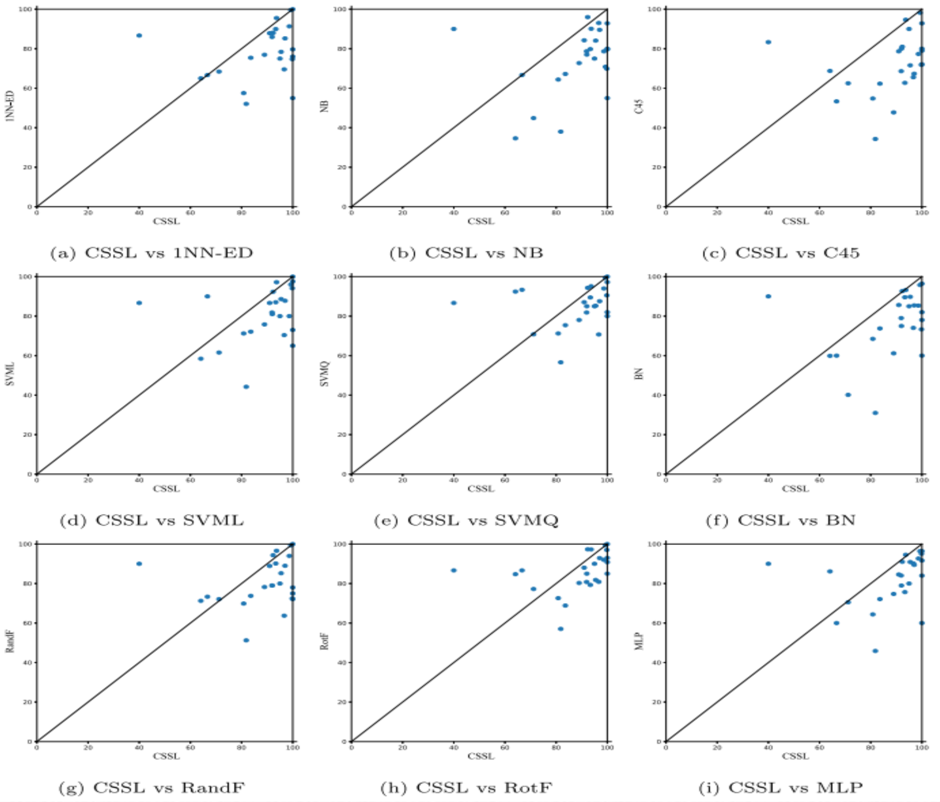
Table 3
The average number of shapelets for each class and the number of calculations of shapelet updating at each iteration.

#	Dataset	$Num_{shapelet}$		$Num_{updating}$	
		LS	CSSL	LS	CSSL
1	Beef	120	4.2	600	21
2	BeetleFly	27	4	27	8
3	BirdChicken	18	2	18	4
4	CBF	34	6	102	18
5	ChlorineConcentration	69	9	207	27
6	Coffee	27	6	27	12
7	ECG200	27	8	27	16
8	ECGFiveDays	24	4	24	8
9	FaceFour	87	15	348	60
10	FacesUCR	420	14.3	5,880	200
11	Gun_Point	27	4	27	8
12	ItalyPowerDemand	24	4	24	8
13	Lighting2	33	5	33	10
14	Lighting7	195	15.7	1,365	110
15	MedicalImages	196	12	1,960	120
16	MoteStrain	24	6	24	12
17	OliveOil	93	1.75	372	7
18	OSULeaf	181	13	1,086	78
19	SonyAIBORobotSurface	24	5.5	24	11
20	SonyAIBORobotSurfaceII	24	6.5	24	13
21	Symbols	96	7.3	576	44
22	SyntheticControl	156	5.3	936	32
23	Trace	64	6.25	256	25
24	TwoLeadECG	14	3.5	14	7
25	Wafer	36	6.5	36	13

Classification accuracy (standard classifiers)

Table 4
Accuracy against standard classifiers.

#	Dataset	1NN-ED	NB	C45	SVML	SVMQ	BN	RandF	RotF	MLP	CSSL (Ours)
1	Beef	66.667	66.667	53.333	90.000	**93.333**	60.000	73.333	86.667	60.000	66.667
2	BeetleFly	75.000	75.000	90.000	80.000	85.000	85.000	80.000	90.000	80.000	**95.000**
3	BirdChicken	55.000	55.000	80.000	65.000	80.000	60.000	75.000	85.000	60.000	**100.000**
4	CBF	85.222	89.556	67.333	87.778	87.556	85.444	89.000	92.889	89.444	**97.000**
5	ChlorineConcentration	65.000	34.635	68.750	58.438	**92.422**	59.896	71.250	84.740	86.146	64.089
6	Coffee	**100.000**	92.857	92.857	**100.000**	**100.000**	96.429	**100.000**	**100.000**	96.429	**100.000**
7	ECG200	88.000	77.000	80.000	81.000	85.000	75.000	79.000	85.000	79.000	**92.000**
8	ECGFiveDays	79.675	79.675	72.125	97.561	97.213	78.049	72.125	90.825	91.638	**100.000**
9	FaceFour	78.409	84.091	71.591	88.636	85.227	89.773	85.227	81.818	90.909	**95.455**
10	FacesUCR	76.927	72.732	47.756	75.805	78.049	61.171	78.244	80.293	74.732	**88.976**
11	GunPoint	91.333	78.667	77.333	80.000	94.000	85.333	94.000	92.000	92.667	**98.667**
12	ItalyPowerDemand	95.530	90.087	94.655	97.182	95.141	93.197	96.599	**97.279**	94.558	93.683
13	Lightning2	75.410	67.213	62.295	72.131	75.410	73.770	73.770	68.852	72.131	**83.607**
14	Lightning7	57.534	64.384	54.795	71.233	71.233	68.493	69.863	72.603	64.384	**80.822**
15	MedicalImages	68.421	44.868	62.500	61.579	70.789	40.132	72.105	**77.237**	70.526	71.184
16	MoteStrain	87.859	84.265	78.674	86.661	87.061	85.623	88.898	88.019	84.585	**90.974**
17	OliveOil	86.667	**90.000**	83.333	86.667	86.667	**90.000**	**90.000**	86.667	**90.000**	40.000
18	OSULeaf	52.066	38.017	34.298	44.215	56.612	30.992	51.240	57.025	45.868	**81.818**
19	SonyAIBORobotSurface1	69.551	93.012	65.557	70.383	70.715	74.043	63.727	80.865	90.183	**96.672**
20	SonyAIBORobotSurface2	85.939	78.699	68.625	81.847	81.847	79.014	79.014	80.797	84.050	**91.920**
21	Symbols	89.950	79.799	62.714	87.035	89.447	89.548	90.151	79.296	75.678	**93.367**
22	SyntheticControl	88.000	96.000	81.000	92.333	94.333	92.667	94.333	**97.333**	91.000	92.333
23	Trace	76.000	80.000	79.000	73.000	82.000	82.000	78.000	93.000	84.000	**100.000**
24	TwoLeadECG	74.715	69.886	71.817	94.118	90.518	73.310	72.432	97.015	95.083	**99.824**
25	Wafer	**99.546**	70.847	98.199	95.993	99.416	95.766	99.351	99.448	96.398	99.270
	Performing Best	2	1	0	1	3	1	2	4	1	**18**
	Percentage (%)	6.061	3.030	0.000	3.030	9.091	3.030	6.061	12.121	3.030	**54.545**
	Average Rank	5.84	7.52	8.32	5.76	4.00	6.88	4.94	3.52	5.58	**2.64**
	Wilcoxon Test p-value	0.001	0.000	0.000	0.004	0.037	0.000	0.005	0.072	0.002	—

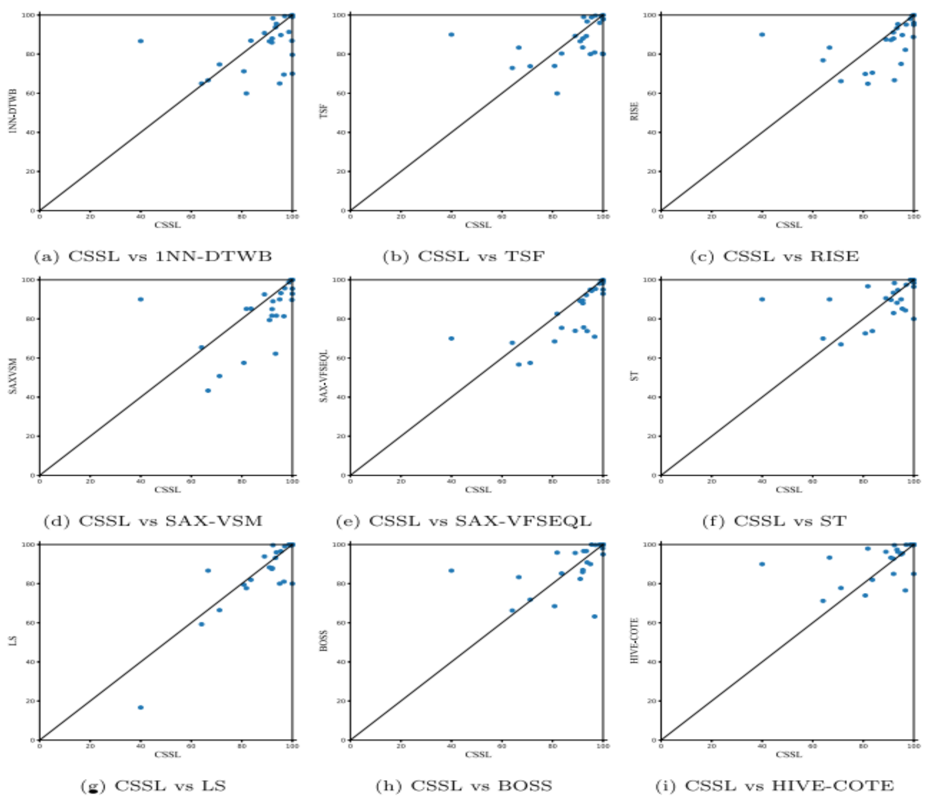


(a) CSSL vs 1NN-ED (b) CSSL vs NB (c) CSSL vs C45
(d) CSSL vs SVML (e) CSSL vs SVMQ (f) CSSL vs BN
(g) CSSL vs RandF (h) CSSL vs RotF (i) CSSL vs MLP

Classification accuracy (state-of-the-art classifiers)

Table 5
Accuracy against state-of-the-art time-series classifiers.

#	Dataset	1NN-DTWB	TSF	RISE	SAX-VSM	SAX-VFSEQL	ST	LS	BOSS	HIVE-COTE	CSSL (Ours)
1	Beef	66.667	83.333	83.333	43.333	56.667	90.000	86.667	83.333	**93.333**	66.667
2	BeetleFly	65.000	80.000	75.000	90.000	**95.000**	90.000	80.000	90.000	**95.000**	**95.000**
3	BirdChicken	70.000	80.000	95.000	**100.000**	95.000	80.000	80.000	95.000	85.000	**100.000**
4	CBF	99.444	99.667	95.111	95.667	95.333	97.444	99.111	99.778	**99.889**	97.000
5	ChlorineConcentration	65.000	72.917	**76.849**	65.443	67.786	69.974	59.245	66.276	71.198	64.089
6	Coffee	**100.000**	**100.000**	**100.000**	92.857	92.857	96.429	**100.000**	**100.000**	**100.000**	**100.000**
7	ECG200	88.000	88.000	88.000	85.000	88.000	83.000	88.000	87.000	85.000	**92.000**
8	ECGFiveDays	79.675	97.793	99.884	95.470	99.187	98.374	**100.000**	**100.000**	**100.000**	**100.000**
9	FaceFour	89.773	98.864	89.773	93.182	94.318	85.227	96.591	**100.000**	95.455	95.455
10	FacesUCR	90.780	89.317	87.512	92.537	73.902	90.585	93.902	95.707	**96.293**	88.976
11	GunPoint	91.333	96.000	98.000	98.667	98.000	**100.000**	**100.000**	**100.000**	**100.000**	98.667
12	ItalyPowerDemand	95.530	**96.696**	95.335	81.633	73.761	94.752	96.016	90.865	96.307	93.683
13	Lightning2	**86.885**	80.328	70.492	85.246	75.410	73.770	81.967	85.246	81.967	83.607
14	Lightning7	71.233	73.973	69.863	57.534	68.493	72.603	79.452	68.493	73.973	**80.822**
15	MedicalImages	74.737	73.816	66.184	50.789	57.500	66.974	66.447	71.842	**77.763**	71.184
16	MoteStrain	86.581	86.581	87.220	79.393	89.137	89.696	88.339	82.428	**93.291**	90.974
17	OliveOil	86.667	**90.000**	**90.000**	**90.000**	70.000	**90.000**	16.667	86.667	**90.000**	40.000
18	OSULeaf	59.917	59.917	64.876	85.124	82.645	96.694	77.686	95.868	**97.934**	81.818
19	SonyAIBORobotSurface1	69.551	80.865	82.196	81.364	70.882	84.359	81.032	63.228	76.539	**96.672**
20	SonyAIBORobotSurface2	85.939	83.421	91.081	81.637	89.507	**93.389**	87.513	85.939	92.760	91.920
21	Symbols	93.769	89.246	93.266	62.211	92.161	88.241	93.166	96.683	**97.387**	93.367
22	SyntheticControl	98.333	99.000	66.667	89.000	75.667	98.333	**99.667**	96.667	**99.667**	92.333
23	Trace	99.000	98.000	96.000	**100.000**	**100.000**	**100.000**	**100.000**	**100.000**	**100.000**	**100.000**
24	TwoLeadECG	86.831	80.246	88.762	89.728	98.156	99.737	99.649	98.068	99.649	**99.824**
25	Wafer	99.594	99.530	99.546	99.903	99.562	**100.000**	99.611	99.481	99.935	99.270
	Performing Best	2	3	3	3	2	5	5	5	**14**	9*
	Percentage (%)	3.922	5.882	5.882	5.882	3.922	9.804	9.804	9.804	**27.451**	17.647*
	Average Rank	6.54	5.88	6.44	6.78	6.74	5.04	4.88	5.08	**2.96**	4.66*
	Wilcoxon Test p-value	0.068	0.219	0.024	0.007	0.003	0.511	0.158	0.910	0.117	—



(a) CSSL vs 1NN-DTWB
(b) CSSL vs TSF
(c) CSSL vs RISE
(d) CSSL vs SAX-VSM
(e) CSSL vs SAX-VFSEQL
(f) CSSL vs ST
(g) CSSL vs LS
(h) CSSL vs BOSS
(i) CSSL vs HIVE-COTE

Brief view on interpretability

CSSL is highly interpretable because the class-specific shapelets represent the distinguishing shape characteristics of one class, such as a fluctuation within a sensor reading reflecting a specific physical process. Thus, the more the shape characteristics of a time series are similar to the class-specific shapelets of a class, the more likely the time series belongs to the class.

In more general cases where one class cannot be simply distinguished against the others by one shapelet, a set of **class specific shapelets in one class are jointly used to transform the time series into a high dimensional space** based on the distances from the time series to all these shapelets, where two classes can be maximally separated.

Furthermore, the **classification decision of CSSL is easier to understand compared to other state-of-the-art classifiers** (such as whole-series-based methods or interval-based methods).

Conclusions

In this paper, they built a **more efficient and interpretable time-series classification** while guaranteeing a competitive accuracy by proposing a novel shapelet learning algorithm called CSSL. To reduce the number of calculations of shapelet updating:

- They proposed to **learn class-specific shapelets that can maximally distinguish a class** against others.
- Then, they developed a **class specific shapelet candidates discovery method to automatically determine the number, length, and initial values of the shapelets** in the learning model to further reduce the number of shapelets to be updated at each iteration and the number of iterations.

The CSSL method achieves great results in terms of speedup against LS and performs well against state-of-the-art models while being more interpretable.