Scalable and Accurate Subsequence Transform for Time Series Classification

Michael F. MBOUOPDA Engelbert MEPHU NGUIFO

University of Clermont Auvergne, LIMOS, UMR 6158, CNRS,
Clermont-Ferrand, France

Conférence sur l'Apprentissage Automatique
14-16 June 2021, Saint-Etienne, France

Table of contents

Table of Contents

Table of Contents

Motivation

- Time series classification (TSC) has gained a lot of interest during the last decade.

Motivation

- Time series classification (TSC) has gained a lot of interest during the last decade.
- Diverse range of applications such as galaxy and stars classification, sleep detection, power consumption analysis, pathogen identification, etc

Motivation

- Time series classification (TSC) has gained a lot of interest during the last decade.
- Diverse range of applications such as galaxy and stars classification, sleep detection, power consumption analysis, pathogen identification, etc
- Many effective methods (HIVE-COTE, ROCKET, TS-CHIEF, STC, etc) but none is scalable, accurate and interpretable.

Motivation

- Time series classification (TSC) has gained a lot of interest during the last decade.
- Diverse range of applications such as galaxy and stars classification, sleep detection, power consumption analysis, pathogen identification, etc
- Many effective methods (HIVE-COTE, ROCKET, TS-CHIEF, STC, etc) but none is scalable, accurate and interpretable.
- Classification using *shapelet* features is interpretable, accurate but not *scalable*

Motivation

- Time series classification (TSC) has gained a lot of interest during the last decade.
- Diverse range of applications such as galaxy and stars classification, sleep detection, power consumption analysis, pathogen identification, etc
- Many effective methods (HIVE-COTE, ROCKET, TS-CHIEF, STC, etc) but none is scalable, accurate and interpretable.
- Classification using *shapelet* features is interpretable, accurate but not *scalable*

Goal

Build a shapelet method that is **accurate**, **interpretable** and **scalable**

Table of Contents

Basic notions

- **Time series** of length m: $T = (t_1, t_2, ..., T_m), t_i \in \mathbb{R}$

Basic notions

- **Time series** of length m: $T = (t_1, t_2, ..., T_m), t_i \in \mathbb{R}$
- **Subsequence** of length l in T:
  $S = (s_1, s_2, ..., s_l) = (t_j, t_{j+1}, ..., t_{j+l})$

Basic notions

- **Time series** of length m: $T = (t_1, t_2, ..., T_m), t_i \in \mathbb{R}$
- **Subsequence** of length l in T:
  $S = (s_1, s_2, ..., s_l) = (t_j, t_{j+1}, ..., t_{j+l})$
- **Distance** between S and T:

$$dist(T, S) = \min_{\forall R \in W_l}(\{\sum_{i=1}^{l}(r_i - s_i)^2\})$$

Basic notions

- **Time series** of length m: $T = (t_1, t_2, ..., T_m), t_i \in \mathbb{R}$
- **Subsequence** of length l in T:
  $S = (s_1, s_2, ..., s_l) = (t_j, t_{j+1}, ..., t_{j+l})$
- **Distance** between S and T:

$$dist(T, S) = \min_{\forall R \in W_l}(\{\sum_{i=1}^{l}(r_i - s_i)^2\})$$

- **Separator**: a subsequence S that divides a time series dataset in two groups.

Basic notions

- **Time series** of length m: $T = (t_1, t_2, ..., T_m), t_i \in \mathbb{R}$
- **Subsequence** of length l in T:
  $S = (s_1, s_2, ..., s_l) = (t_j, t_{j+1}, ..., t_{j+l})$
- **Distance** between S and T:

$$dist(T, S) = \min_{\forall R \in W_l}(\{\sum_{i=1}^{l}(r_i - s_i)^2\})$$

- **Separator**: a subsequence S that divides a time series dataset in two groups.
- **Shapelet**: a separator that maximizes the information gain

Basic notions

- **Time series** of length m: $T = (t_1, t_2, ..., T_m), t_i \in \mathbb{R}$
- **Subsequence** of length l in T:
  $S = (s_1, s_2, ..., s_l) = (t_j, t_{j+1}, ..., t_{j+l})$
- **Distance** between S and T:

$$dist(T, S) = \min_{\forall R \in W_l}(\{\sum_{i=1}^{l}(r_i - s_i)^2\})$$

- **Separator**: a subsequence S that divides a time series dataset in two groups.
- **Shapelet**: a separator that maximizes the information gain

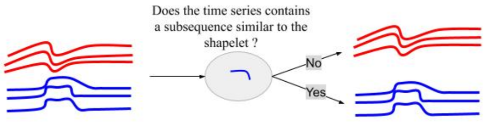

Fig. 1: Shapelet illustration

Overview of Shapelet Transform Classification [Hills et al., 2014]

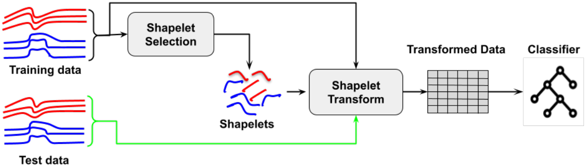

Fig. 2: Overview of Shapelet Transform Classification

Overview of Shapelet Transform Classification [Hills et al., 2014]

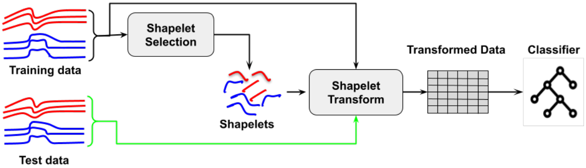

Fig. 2: Overview of Shapelet Transform Classification

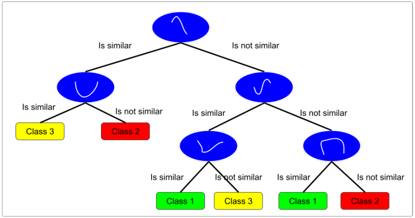

Fig. 3: A shapelet decision tree that could be obtained after STC training

Overview of Shapelet Transform Classification [Hills et al., 2014]

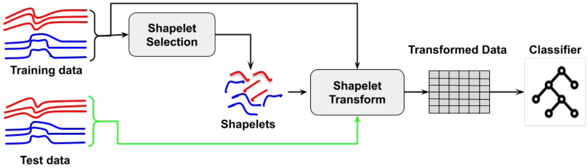

Fig. 2: Overview of Shapelet Transform Classification

Strengths

- Accurate
- Robust to outliers
- Interpretable

Limitations

- Time complexity: $O(n^2m^4)$
- Prone to overfitting

Table of Contents

Inspiration

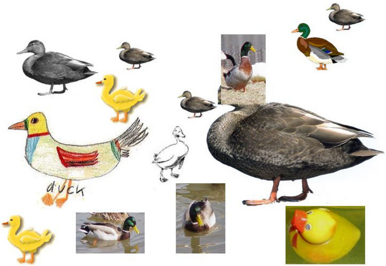

Fig. 3: Illustration of invariance in recognition Heeger [2002-2014]

Definition

Core object recognition is the ability to recognize objects despite substantial appearance variations [DiCarlo et al., 2012]

Inspiration

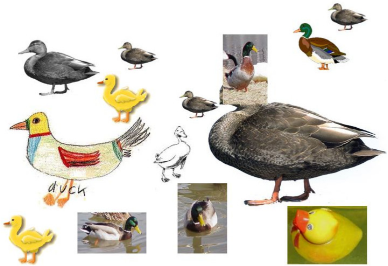

Fig. 3: Illustration of invariance in recognition Heeger [2002-2014]

Definition

Core object recognition is the ability to recognize objects despite substantial appearance variations [DiCarlo et al., 2012]

From *core object recognition* to *core shapelet recognition*

Definition

Core shapelet recognition is the ability to recognize any variant of a shapelet from one or a few number of its variants

From *core object recognition* to *core shapelet recognition*

Definition

Core shapelet recognition is the ability to recognize any variant of a shapelet from one or a few number of its variants

Our claims for building a core shapelet recognition-based time series classifier:

1. It should not be necessary to assess a lot of variants of a shapelet candidate

From *core object recognition* to *core shapelet recognition*

Definition

Core shapelet recognition is the ability to recognize any variant of a shapelet from one or a few number of its variants

Our claims for building a core shapelet recognition-based time series classifier:

1. It should not be necessary to assess a lot of variants of a shapelet candidate
2. Any single time series should *contain all the shapelets* of its corresponding class.

From *core object recognition* to *core shapelet recognition*

Definition

Core shapelet recognition is the ability to recognize any variant of a shapelet from one or a few number of its variants

Our claims for building a core shapelet recognition-based time series classifier:

1. It should not be necessary to assess a lot of variants of a shapelet candidate
2. Any single time series should *contain all the shapelets* of its corresponding class.
3. Filtering shapelet candidates beforehand of classification could lead to an inaccurate model

From *core object recognition* to *core shapelet recognition*

Definition

Core shapelet recognition is the ability to recognize any variant of a shapelet from one or a few number of its variants

Our claims for building a core shapelet recognition-based time series classifier:

1. It should not be necessary to assess a lot of variants of a shapelet candidate
2. Any single time series should *contain all the shapelets* of its corresponding class.
3. Filtering shapelet candidates beforehand of classification could lead to an inaccurate model
4. The classifier could automatically learn the best shapelets

Core shapelet recognition

Illustration with the Chinatown dataset

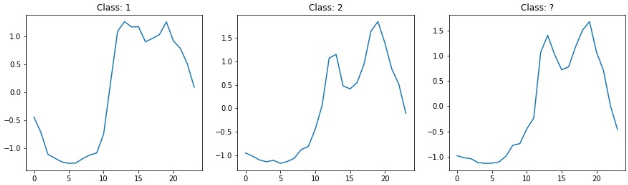

Fig. 4: Three random instances from the Chinatown dataset

Core shapelet recognition

Illustration with the Chinatown dataset

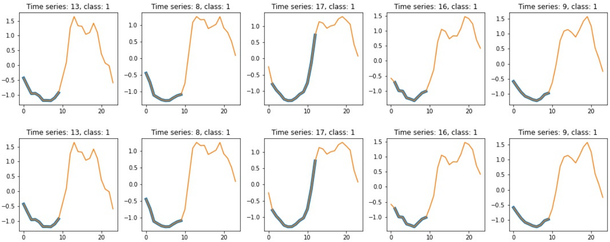

Fig. 4: Shapelet learned by STC. Accuracy = 0.97, Running time = 51 secs

Core shapelet recognition

Illustration with the Chinatown dataset

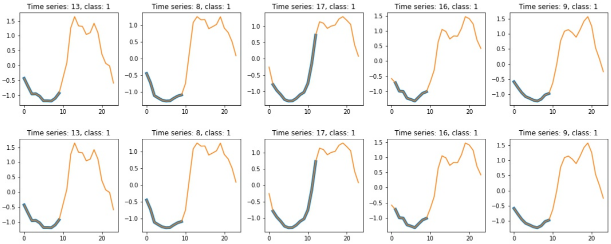

Fig. 4: Shapelet learned by STC. Accuracy = 0.97, Running time = 51 secs

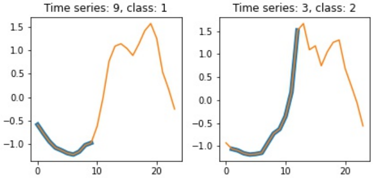

Fig. 5: Shapelet learned by STC-1. Accuracy = 0.96, Running time = 10 secs

SAST: Scalable and Accurate Subsequence Transform

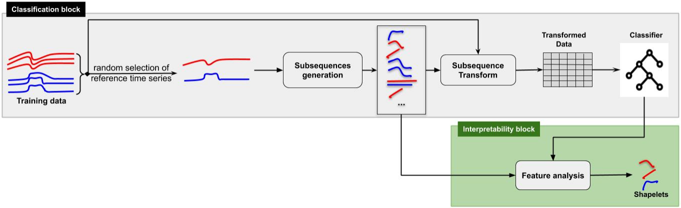

Fig. 6: Overview of time series classification using SAST

SAST: Scalable and Accurate Subsequence Transform

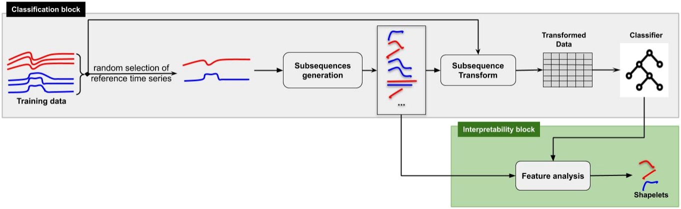

Fig. 6: Overview of time series classification using SAST

Unlike STC, SAST

- Use one instance per class to build the shapelet space

SAST: Scalable and Accurate Subsequence Transform

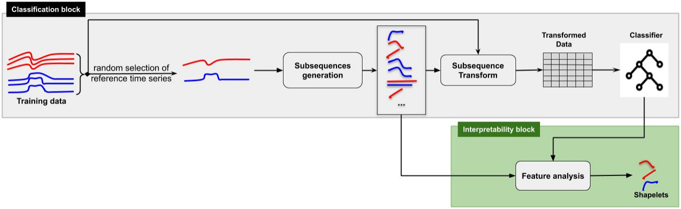

Fig. 6: Overview of time series classification using SAST

Unlike STC, SAST

- Use one instance per class to build the shapelet space
- Does not prune shapelet candidates beforehand of classification

SAST: Scalable and Accurate Subsequence Transform

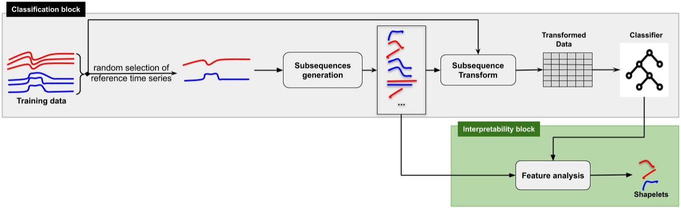

Fig. 6: Overview of time series classification using SAST

Unlike STC, SAST

- Use one instance per class to build the shapelet space
- Does not prune shapelet candidates beforehand of classification
- Has a time complexity $O(nm^3)$

Table of Contents

Setup

Datasets and models

Datasets

- 72 datasets from the UCR & UEA archive

Setup

Datasets and models

Datasets

- 72 datasets from the UCR & UEA archive
- Already split in train/test

Datasets

- 72 datasets from the UCR & UEA archive
- Already split in train/test
- Diverse problem types and properties

Setup

Datasets and models

Datasets

- 72 datasets from the UCR & UEA archive
- Already split in train/test
- Diverse problem types and properties

Models

- Shapelet based
    - STC, STC-k

Setup

Datasets and models

Datasets

- 72 datasets from the UCR & UEA archive
- Already split in train/test
- Diverse problem types and properties

Models

- Shapelet based
  - STC, STC-k
  - ELIS++, LS, FS

Setup

Datasets and models

Datasets

- 72 datasets from the UCR & UEA archive
- Already split in train/test
- Diverse problem types and properties

Models

- Shapelet based
  - STC, STC-k
  - ELIS++, LS, FS
- Non-shapelet based:
  - HIVE-COTE
  - ROCKET
  - TS-CHIEF

Setup

Implementation and parameters

Implementation

- Implemented in Python

Setup

Implementation and parameters

Implementation

- Implemented in Python
- Compatible with Scikit-learn API

Implementation

- Implemented in Python
- Compatible with Scikit-learn API
- Open source: https://github.com/frankl1/sast

Setup

Implementation and parameters

Implementation

- Implemented in Python
- Compatible with Scikit-learn API
- Open source: https://github.com/frankl1/sast

STC, STC-k and SAST parameters

- classifier: Ridge classifier with LOO-CV

Setup

Implementation and parameters

Implementation

- Implemented in Python
- Compatible with Scikit-learn API
- Open source: https://github.com/frankl1/sast

STC, STC-k and SAST parameters

- classifier: Ridge classifier with LOO-CV
- $Shapelet\_length = \{3, 4, ..., m\}$

Setup

Implementation and parameters

Implementation

- Implemented in Python
- Compatible with Scikit-learn API
- Open source: https://github.com/frankl1/sast

STC, STC-k and SAST parameters

- classifier: Ridge classifier with LOO-CV
- $Shapelet\_length = \{3, 4, ..., m\}$
- $k = \{1, 0.25, 0.5, 0.75\}$, always 1 for SAST

Setup

Implementation and parameters

Implementation

- Implemented in Python
- Compatible with Scikit-learn API
- Open source: https://github.com/frankl1/sast

STC, STC-k and SAST parameters

- classifier: Ridge classifier with LOO-CV
- $Shapelet\_length = \{3, 4, ..., m\}$
- $k = \{1, 0.25, 0.5, 0.75\}$, always 1 for SAST
- STC and STC-k time contract: 1 *hour*

Table of Contents

STC-k accuracy results

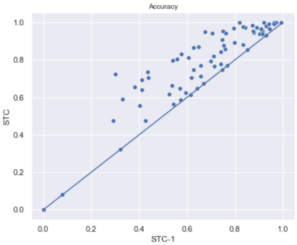

Fig. 7: STC(69 wins) vs STC-1 (1 win), 2 draws

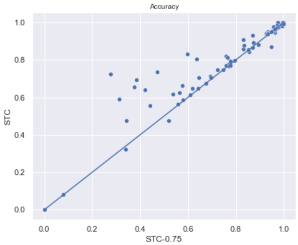

Fig. 8: STC(50 wins) vs STC-0.75 (16 win), 6 draws

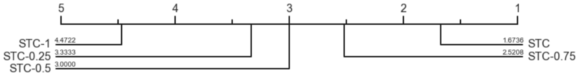

Fig. 9: Critical difference diagram between STC-k models

SAST vs STC

Fig. 10: SAST(66 wins) vs STC-1 (5 win), 1 draw

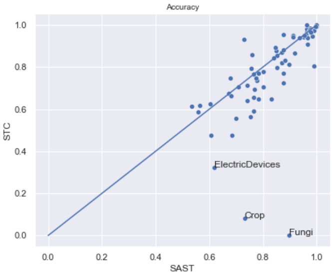

Fig. 11: SAST(43 wins) vs STC (27 win), 2 draws

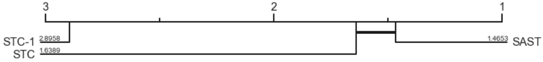

Fig. 12: Critical difference diagram between SAST, STC and STC-k

SAST vs other shapelet methods

The 35 datasets used in the ELIS++ paper

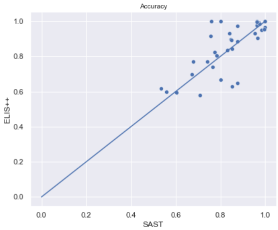

Fig. 13: SAST(12 wins) vs ELIS++ (22 win), 1 draw

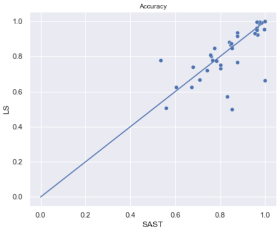

Fig. 14: SAST(17 wins) vs LS (16 win), 2 draws

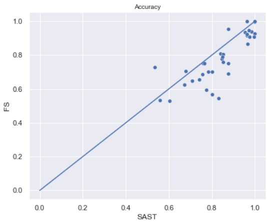

Fig. 15: SAST(30 wins) vs FS (4 win), 1 draw

SAST vs other shapelet methods

The 35 datasets used in the ELIS++ paper

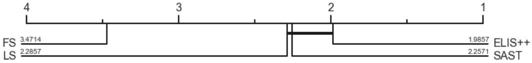

Fig. 16: Critical difference diagram between SAST and other shapelet methods

SAST vs other shapelet methods

The 35 datasets used in the ELIS++ paper

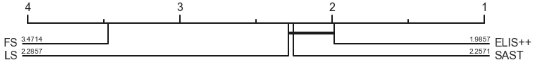

Fig. 16: Critical difference diagram between SAST and other shapelet methods

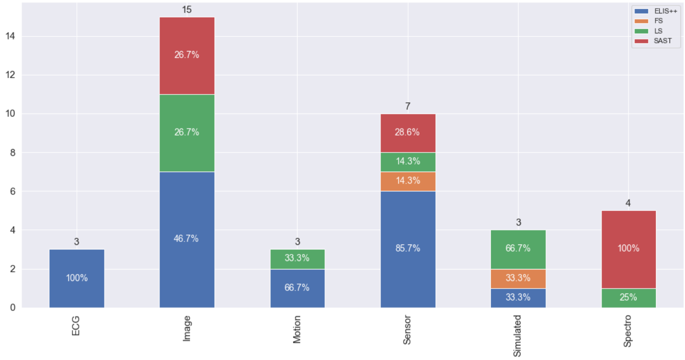

Fig. 17: Win percentage per problem type

SAST vs non-shapelet methods

67 datasets for which results are published on the UEA & UCR archive

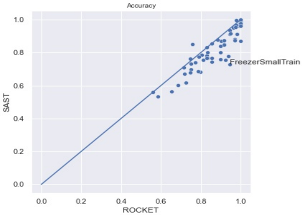

Fig. 18: SAST (5 wins) vs ROCKET (58 wins), 4 draws

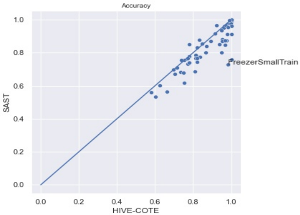

Fig. 19: SAST (10 wins) vs HC (53 win), 4 draws

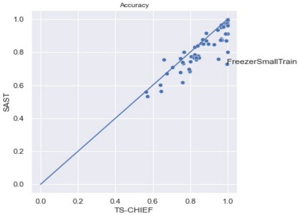

Fig. 20: SAST (9 wins) vs TS-CHIEF (55 win), 3 draws

SAST vs non-shapelet methods

67 datasets for which results are published on the UEA & UCR archive

- Average accuracies are 0.84 ± 0.12 (SAST), 0.88 ± 0.11 (ROCKET), 0.88 ± 0.11 (HC) and 0.88 ± 0.12 (TS-CHIEF)

SAST vs non-shapelet methods

67 datasets for which results are published on the UEA & UCR archive

- Average accuracies are 0.84 ± 0.12 (SAST), 0.88 ± 0.11 (ROCKET), 0.88 ± 0.11 (HC) and 0.88 ± 0.12 (TS-CHIEF)
- No statistical difference among the four models

SAST vs non-shapelet methods

67 datasets for which results are published on the UEA & UCR archive

- Average accuracies are 0.84 ± 0.12 (SAST), 0.88 ± 0.11 (ROCKET), 0.88 ± 0.11 (HC) and 0.88 ± 0.12 (TS-CHIEF)
- No statistical difference among the four models

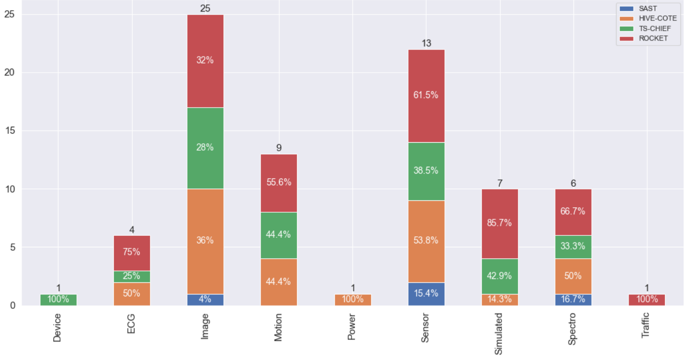

Fig. 21: Win percentage per problem type

Table of Contents

Scalability results

- SASTEN: ensemble of 3 SAST models
- SASTEN-A: ensemble of 3 SAST models, each one working on shapelet length in intervals $[3;9]$, $[10;16]$ and $[17,23]$ respectivelly

Scalability results

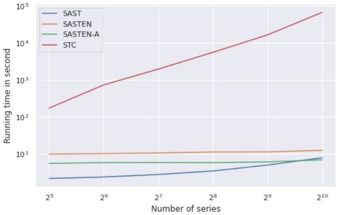

Fig. 22: Regarding the number of time series

Nb of TS	64	1024
SAST	2	7
STC	720	67000
Speedup	$36\times$	$\sim 9500\times$

Scalability results

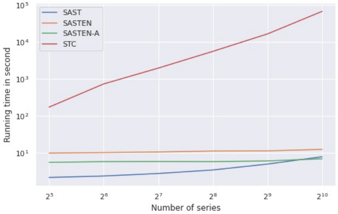

Fig. 22: Regarding the number of time series

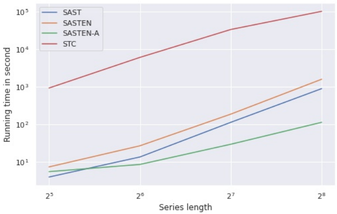

Fig. 23: Regarding the time series length

Nb of TS	64	1024
SAST	2	7
STC	720	67000
Speedup	36×	∼ 9500×

TS length	64	256
SAST	13	892
STC	6016	100585
Speedup	∼ 462×	∼ 112×

Table of Contents

How to explain SAST?

SAST is explained by identifying the top best features (i.e shapelets) learned by the classifier:

How to explain SAST?

SAST is explained by identifying the top best features (i.e shapelets) learned by the classifier:

- For linear models, the subsequences with the highest weight norms

How to explain SAST?

SAST is explained by identifying the top best features (i.e shapelets) learned by the classifier:

- For linear models, the subsequences with the highest weight norms
- For decision tree models, the subsequences with the highest information gains

How to explain SAST?

SAST is explained by identifying the top best features (i.e shapelets) learned by the classifier:

- For linear models, the subsequences with the highest weight norms
- For decision tree models, the subsequences with the highest information gains
- In any case, a post-hoc method such as LIME, SHAP, Saliency maps

Top best shapelets learned by SAST on the Chinatown dataset

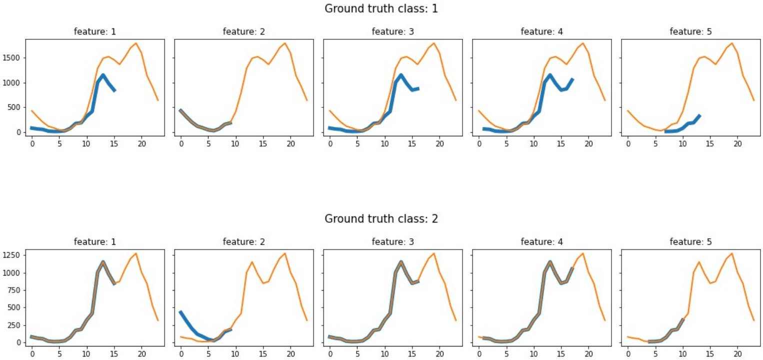

Fig. 24: Top 5 best shapelets plotted on the reference time series

Top best shapelets learned by SAST on the Chinatown dataset

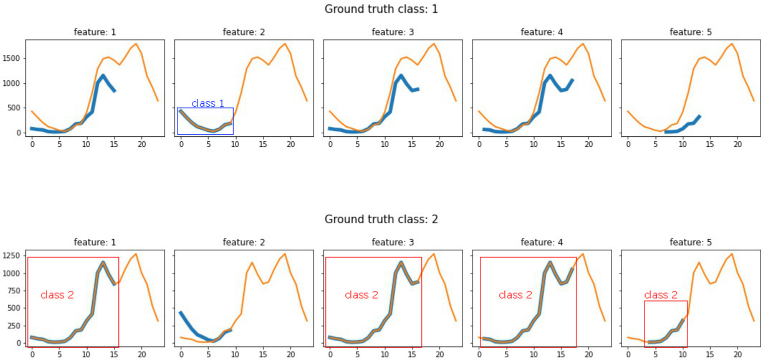

Fig. 24: Top 5 best shapelets plotted on the reference time series

Prediction explanations for two random test instances

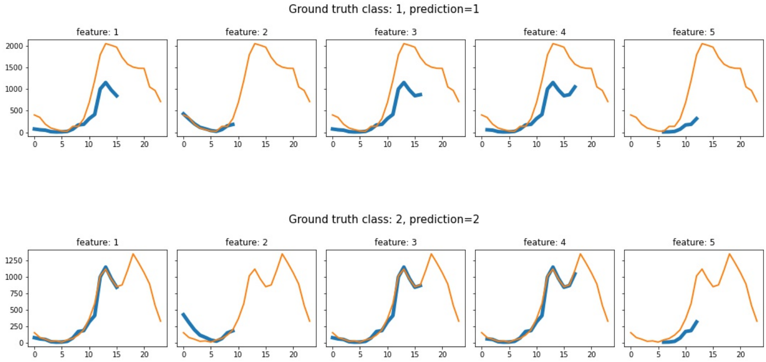

Fig. 25: Explanation for

Prediction explanations for two random test instances

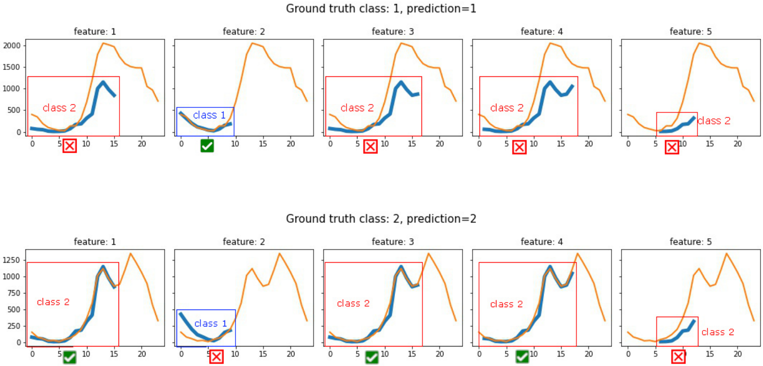

Fig. 25: Top 5 best shapelets plotted on the reference time series

Table of Contents

Conclusion and future direction

- **What we have done**
  - We introduced the core shapelet recognition task, which is the hability to recognize any variant of a shapelet from one or few number of its variants
  - We proposed SAST, which effectively performs the core shapelet recognition task
  - We shown using 72 state of the art datasets that SAST is accurate, interpretable and much more scalable
- **Future directions**
  - Identify and remove duplicate subsequence from SAST
  - Use SAST as a remplacement of STC module in HIVE-COTE
  - Apply the core shapelet recognition task in TS-CHIEF

References

J. J. DiCarlo, D. Zoccolan, and N. C. Rust. How does the brain solve visual object recognition? *Neuron*, 73(3): 415–434, 2012.

D. Heeger. Center for neural science, lecture notes: Perception (undergraduate), 2002-2014. URL: `http://www.cns.nyu.edu/~david/courses/perception/lecturenotes/recognition/recognition.html`. Last visited on May 2021.

J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28(4):851–881, 2014.