cobras^{TS}

A new approach to semi-supervised clustering of time series

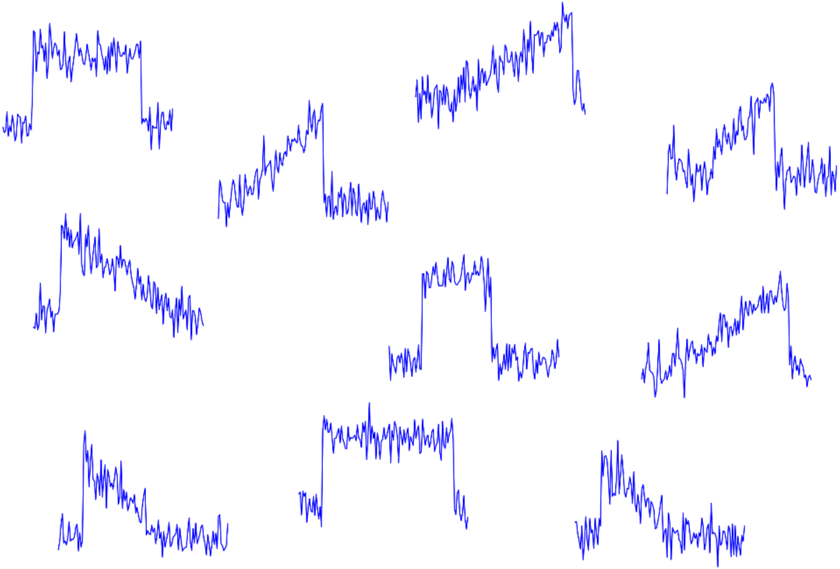
Toon Van Craenendonck, Wannes Meert, Sebastijan Dumancic and Hendrik Blockeel

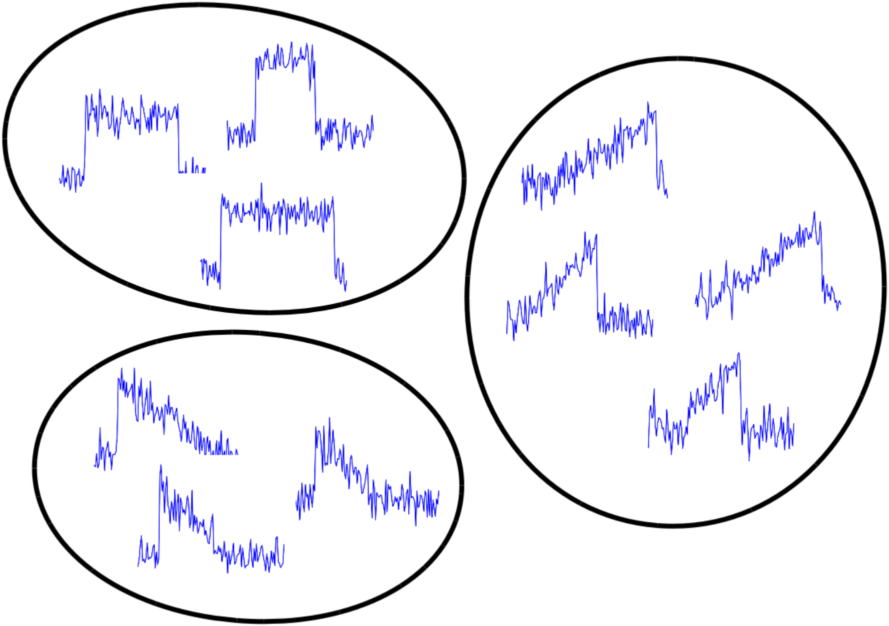
31/10/2018

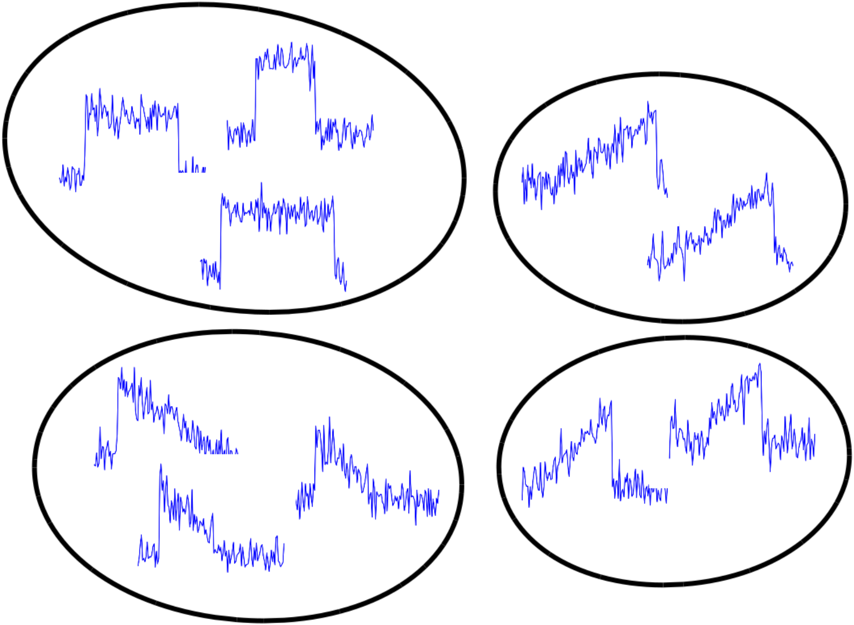
Overview

1. Motivation & background
   - Semi-supervised and interactive clustering
   - COBRAS
2. COBRAS^{TS}
3. Experiments
4. Demo
5. Conclusion

1. Motivation & background
    - Semi-supervised and interactive clustering
    - COBRAS
2. COBRAS^{TS}
3. Experiments
4. Demo
5. Conclusion

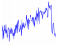
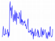


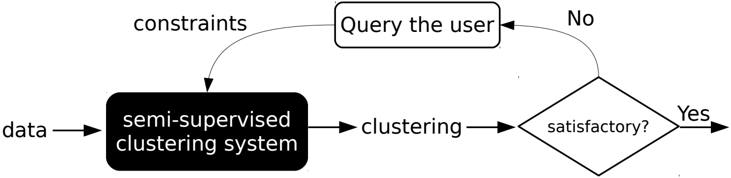


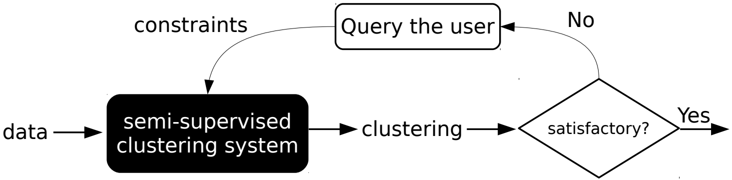


Clustering is **subjective**,
a black box system cannot work

Semi-supervised clustering systems rely on *interaction in the form of pairwise queries*

Should  and  be in the same cluster?





Such an *interactive* workflow requires the clustering system to be:

1. **anytime**: the user can stop and get the best result so far at any time
2. **query-efficient**: few queries before a reasonable result is obtained
3. **time-efficient**: user should not have to wait long between answering queries

COBRAS satisfies these requirements ☺

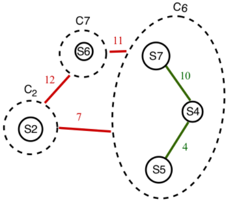
T. Van Craenendonck, S. Dumancic, E. Van Wolputte, H. Blockeel. COBRAS: Interactive clustering with pairwise queries. IDA 2018

T. Van Craenendonck, S. Dumancic, H. Blockeel. COBRA: A fast and simple method for active clustering with pairwise constraints. IJCAI 2017

1. Motivation & background
    - Semi-supervised and interactive clustering
    - COBRAS
2. COBRAS$^{\text{TS}}$
3. Experiments
4. Demo
5. Conclusion

Two key ideas in COBRAS

1. COBRAS uses **super-instances**
   = sets of instances that are temporarily assumed to belong to the same cluster
   = *intermediate level between instances and clusters*

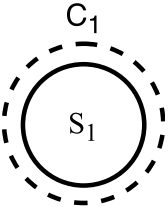


2. It **dynamically refines** these super-instances during clustering

COBRAS pseudocode & example

Input: $\mathcal{X}$: a dataset, q: a query limit
Output: $\mathcal{C}$: a clustering of D

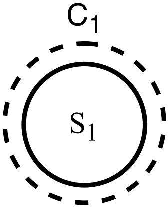
$ML = \emptyset, CL = \emptyset$
$S = \{\mathcal{X}\}, C = \{S\}, \mathcal{C} = \{C\}$
while $|ML| + |CL| < q$ **do**
  $S_{split}, C_{origin} = \arg\max_{S \in C, C \in \mathcal{C}} |S|$
  $k, ML, CL =$ $\texttt{determineSplitLevel}(S_{split}, ML, CL)$
  $S_{new_1}, \ldots, S_{new_k} = \texttt{K-means}(S_{split}, k)$
  $C_{origin} = C_{origin} \setminus \{S_{split}\}$
  $\mathcal{C} = \mathcal{C} \cup \{\{S_{new_1}\}, \ldots, \{S_{new_k}\}\}$
  $\mathcal{C}, ML, CL = \texttt{COBRA}(\mathcal{C}, ML, CL)$
end while
return $\mathcal{C}$



COBRAS pseudocode & example

Input: $\mathcal{X}$: a dataset, q: a query limit
Output: $\mathcal{C}$: a clustering of D

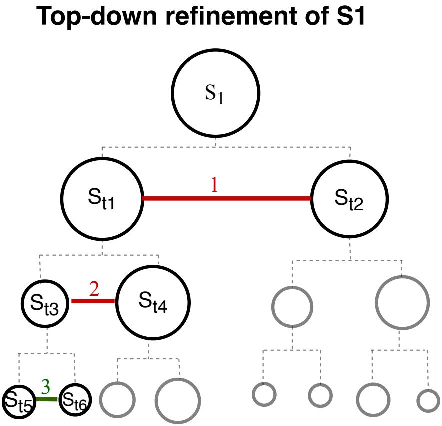
$ML = \emptyset, CL = \emptyset$
$S = \{\mathcal{X}\}, C = \{S\}, \mathcal{C} = \{C\}$
while $|ML| + |CL| < q$ **do**
  $S_{split}, C_{origin} = \arg\max_{S \in C, C \in \mathcal{C}} |S|$
  $k, ML, CL =$ $\texttt{determineSplitLevel}(S_{split}, ML, CL)$
  $S_{new_1}, \ldots, S_{new_k} = \texttt{K-means}(S_{split}, k)$
  $C_{origin} = C_{origin} \setminus \{S_{split}\}$
  $\mathcal{C} = \mathcal{C} \cup \{\{S_{new_1}\}, \ldots, \{S_{new_k}\}\}$
  $\mathcal{C}, ML, CL = \texttt{COBRA}(\mathcal{C}, ML, CL)$
end while
return $\mathcal{C}$



COBRAS pseudocode & example

Input: $\mathcal{X}$: a dataset, q: a query limit
Output: $\mathcal{C}$: a clustering of D

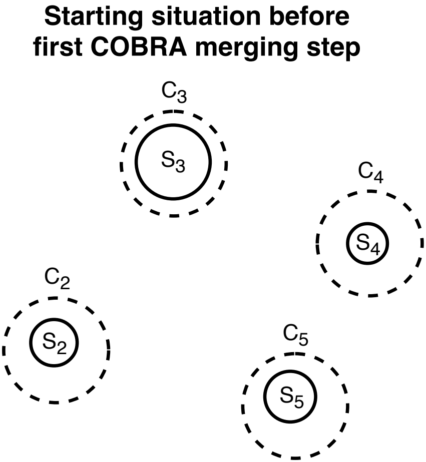
$ML = \emptyset, CL = \emptyset$
$S = \{\mathcal{X}\}, C = \{S\}, \mathcal{C} = \{C\}$
while $|ML| + |CL| < q$ **do**
    $S_{split}, C_{origin} = \arg\max_{S \in C, C \in \mathcal{C}} |S|$
    $k, ML, CL =$ $\texttt{determineSplitLevel}(S_{split}, ML, CL)$
    $S_{new_1}, \ldots, S_{new_k} = \texttt{K-means}(S_{split}, k)$
    $C_{origin} = C_{origin} \setminus \{S_{split}\}$
    $\mathcal{C} = \mathcal{C} \cup \{\{S_{new_1}\}, \ldots, \{S_{new_k}\}\}$
    $\mathcal{C}, ML, CL = \texttt{COBRA}(\mathcal{C}, ML, CL)$
end while
return $\mathcal{C}$



COBRAS pseudocode & example

Input: $\mathcal{X}$: a dataset, q: a query limit
Output: $\mathcal{C}$: a clustering of D

$ML = \emptyset, CL = \emptyset$
$S = \{\mathcal{X}\}, C = \{S\}, \mathcal{C} = \{C\}$
while $|ML| + |CL| < q$ **do**
    $S_{split}, C_{origin} = \arg\max_{S \in C, C \in \mathcal{C}} |S|$
    $k, ML, CL =$
    `determineSplitLevel`($S_{split}, ML,$ (
    $S_{new_1}, \ldots, S_{new_k} =$ `K-means`($S_{split},$
    $C_{origin} = C_{origin} \setminus \{S_{split}\}$
    $\mathcal{C} = \mathcal{C} \cup \{\{S_{new_1}\}, \ldots, \{S_{new_k}\}\}$
    $\mathcal{C}, ML, CL =$ `COBRA`($\mathcal{C}, ML, CL$)
end while
return $\mathcal{C}$

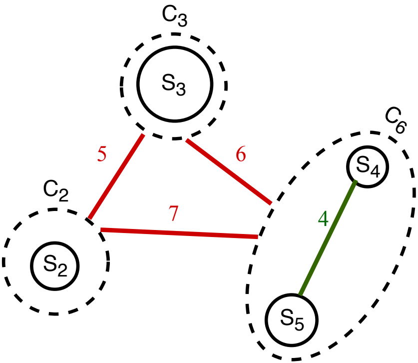


COBRAS pseudocode & example

Input: $\mathcal{X}$: a dataset, q: a query limit
Output: $\mathcal{C}$: a clustering of D

$ML = \emptyset, CL = \emptyset$
$S = \{\mathcal{X}\}, C = \{S\}, \mathcal{C} = \{C\}$
while $|ML| + |CL| < q$ **do**
    $S_{split}, C_{origin} = \arg\max_{S \in C, C \in \mathcal{C}} |S|$
    $k, ML, CL =$ `determineSplitLevel`$(S_{split}, ML, C$
    $S_{new_1}, \ldots, S_{new_k} =$ `K-means`$(S_{split}, k$
    $C_{origin} = C_{origin} \setminus \{S_{split}\}$
    $\mathcal{C} = \mathcal{C} \cup \{\{S_{new_1}\}, \ldots, \{S_{new_k}\}\}$
    $\mathcal{C}, ML, CL =$ `COBRA`$(\mathcal{C}, ML, CL)$
end while
return $\mathcal{C}$

After first bottom up COBRA merging step

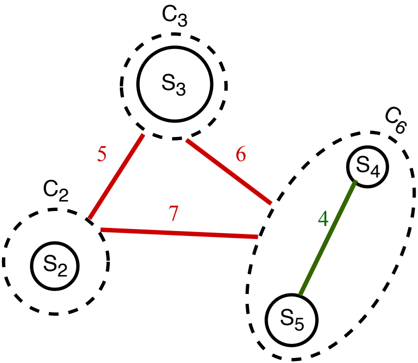


COBRAS pseudocode & example

Input: $\mathcal{X}$: a dataset, q: a query limit
Output: $\mathcal{C}$: a clustering of D

```
ML = ∅, CL = ∅
S = {X}, C = {S}, 𝒞 = {C}
while |ML| + |CL| < q do
    S_split, C_origin = arg max_{S∈C, C∈𝒞} |S|
    k, ML, CL =
    determineSplitLevel(S_split, ML, C
    S_new_1, ..., S_new_k = K-means(S_split, k
    C_origin = C_origin \ {S_split}
    𝒞 = 𝒞 ∪ {{S_new_1}, ..., {S_new_k}}
    𝒞, ML, CL = COBRA(𝒞, ML, CL)
end while
return 𝒞
```

After first bottom up COBRA merging step

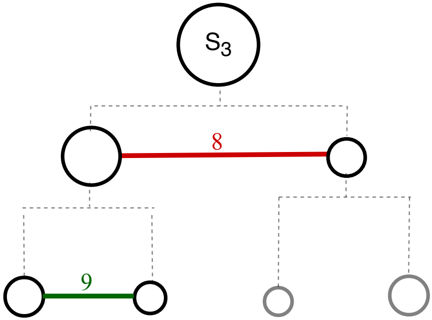


COBRAS pseudocode & example

Input: $\mathcal{X}$: a dataset, q: a query limit
Output: $\mathcal{C}$: a clustering of D

$ML = \emptyset, CL = \emptyset$
$S = \{\mathcal{X}\}, C = \{S\}, \mathcal{C} = \{C\}$
while $|ML| + |CL| < q$ **do**
  $S_{split}, C_{origin} = \arg\max_{S \in C, C \in \mathcal{C}} |S|$
  $k, ML, CL =$
  $\texttt{determineSplitLevel}(S_{split}, ML, C$
  $S_{new_1}, \ldots, S_{new_k} = \texttt{K-means}(S_{split}, k$
  $C_{origin} = C_{origin} \setminus \{S_{split}\}$
  $\mathcal{C} = \mathcal{C} \cup \{\{S_{new_1}\}, \ldots, \{S_{new_k}\}\}$
  $\mathcal{C}, ML, CL = \texttt{COBRA}(\mathcal{C}, ML, CL)$
end while
return $\mathcal{C}$

Top-down refinement of S3

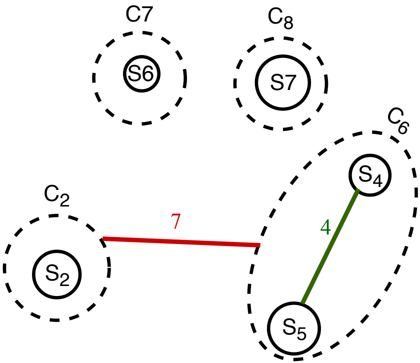


COBRAS pseudocode & example

Input: $\mathcal{X}$: a dataset, q: a query limit
Output: $\mathcal{C}$: a clustering of D

$ML = \emptyset, CL = \emptyset$
$S = \{\mathcal{X}\}, C = \{S\}, \mathcal{C} = \{C\}$
while $|ML| + |CL| < q$ **do**
    $S_{split}, C_{origin} = \arg\max_{S \in C, C \in \mathcal{C}} |S|$
    $k, ML, CL =$ `determineSplitLevel`$(S_{split}, ML, C$
    $S_{new_1}, \ldots, S_{new_k} =$ `K-means`$(S_{split}, k$
    $C_{origin} = C_{origin} \setminus \{S_{split}\}$
    $\mathcal{C} = \mathcal{C} \cup \{\{S_{new_1}\}, \ldots, \{S_{new_k}\}\}$
    $\mathcal{C}, ML, CL =$ `COBRA`$(\mathcal{C}, ML, CL)$
end while
return $\mathcal{C}$

Starting situation before second COBRA merging step

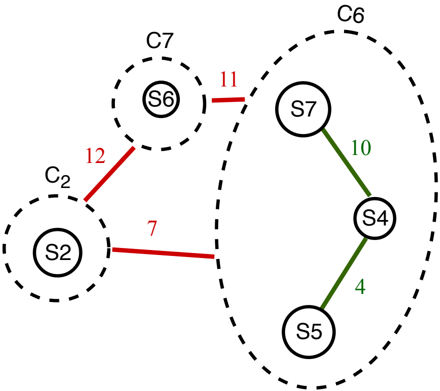


COBRAS pseudocode & example

Input: $\mathcal{X}$: a dataset, q: a query limit
Output: $\mathcal{C}$: a clustering of D

$ML = \emptyset, CL = \emptyset$
$S = \{\mathcal{X}\}, C = \{S\}, \mathcal{C} = \{C\}$
while $|ML| + |CL| < q$ **do**
  $S_{split}, C_{origin} = \arg\max_{S \in C, C \in \mathcal{C}} |S|$
  $k, ML, CL =$
  $\texttt{determineSplitLevel}(S_{split}, ML, C$
  $S_{new_1}, \ldots, S_{new_k} = \texttt{K-means}(S_{split}, k$
  $C_{origin} = C_{origin} \setminus \{S_{split}\}$
  $\mathcal{C} = \mathcal{C} \cup \{\{S_{new_1}\}, \ldots, \{S_{new_k}\}\}$
  $\mathcal{C}, ML, CL = \texttt{COBRA}(\mathcal{C}, ML, CL)$
end while
return $\mathcal{C}$

After second bottom up COBRA merging step



Problem: COBRAS is not suited for time series clustering out-of-the-box

1. It defines super-instance medoids w.r.t. the *Euclidean* distance
2. It uses *K-means* to refine super-instances

$\rightarrow$ both of these are sub-state-of-the-art for time series clustering

Solution: Upgrade COBRAS to use distance measure and clustering method suitable for time series

$\rightarrow$ We refer to this approach as $\mathbf{COBRAS^{TS}}$

We propose two instantiations of COBRAS^{TS}

COBRAS$^{\textbf{kShape}}$

COBRAS with the following substitutions:

Euclidean distance $\rightarrow$ shape-based distance
k-means $\rightarrow$ k-Shape

COBRAS$^{\textbf{DTW}}$

Input: X: a dataset
w: the DTW warping window width
γ: kernel width for converting distances to similarities
Output: A clustering

1. Compute the full pairwise DTW distance matrix of X
2. Convert each distance d to an affinity a: $a_{i,j} = e^{-\gamma d_{i,j}}$
3. Run COBRAS, substituting K-means for splitting super-instances with spectral clustering on the previously computed affinity matrix

Experimental setup

- UCR time series archive: 85 datasets from diverse domains

- Classes are assumed to represent the clusters of interest, clusterings are evaluated by computing the adjusted Rand index (ARI) on the test set in 10-fold CV

- Comparing COBRAS^{TS} to

  cDTW^{SS}: Uses constraints to select window width w in cDTW

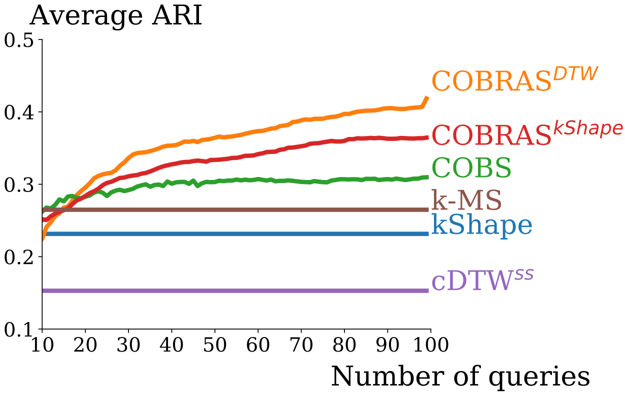
  "Choosing w is critical, and dwarfs any effect of the choice of algorithm."

  COBS: Uses constraints to select and tune an unsupervised algorithm

  k-Shape: k-means variant for time series

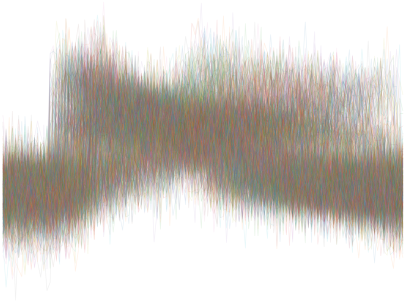
  k-MultiShape: similar to k-Shape, but with *multiple* centroids per cluster; state-of-the-art for unsupervised time series clustering

Experimental results

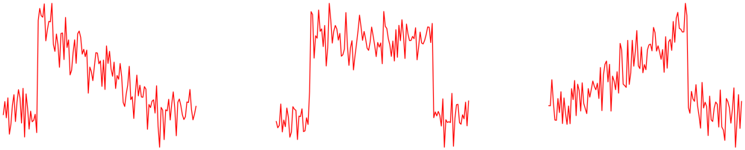


Answering pairwise queries
can substantially improve clustering quality!

Clustering the CBF dataset

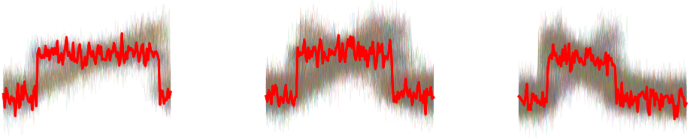


We are interested in separating downward, horizontal and upward patterns

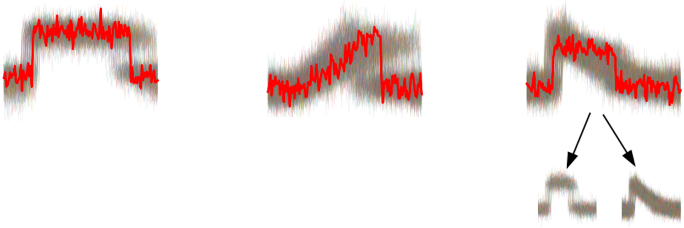


Clustering the CBF dataset

Clustering obtained with cDTW^{SS}

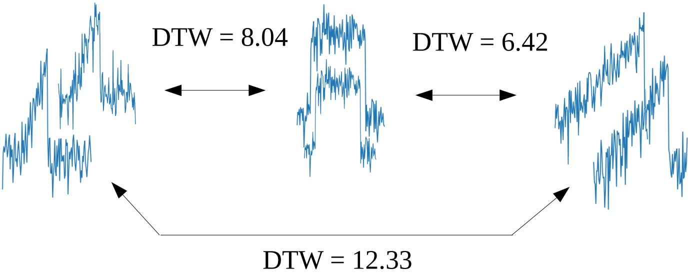


Clustering obtained with kShape



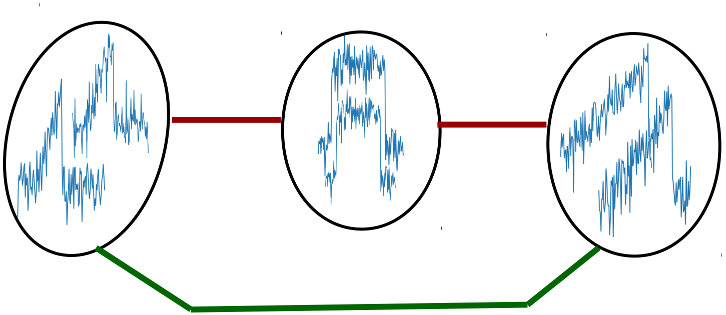
Specifically for CBF, we found an additional reason why COBRAS^{TS} outperforms other systems.

Clustering CBF is difficult as it contains clusters with *separated components*. Existing algorithms cannot deal with this.

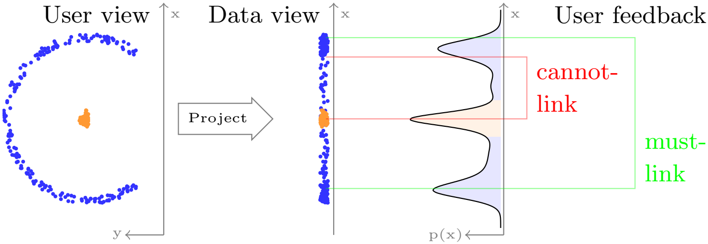


Separated components

This is not a problem for COBRAS^{TS}: components are captured by super-instances, which are grouped into the same cluster through a must-link constraint



Separated components



Coherent clusters may become incoherent when projected onto a subspace

Conclusion

We introduce **COBRAS$^{\mathbf{TS}}$**, an adaptation of COBRAS for time series

Our experimental evaluation shows that:

- **Time series clustering can benefit greatly from small amounts of supervision**, COBRAS^{TS} outperforms competitors *by a large margin*.
- COBRAS^{TS} can detect clusters with **separated components**, and this can be beneficial in time series clustering
- The choice of the clustering algorithm matters (contrary to prior claims)
- COBRAS^{TS} allows the user to **interactively** cluster time series data
  $\rightarrow$ We show this with a concrete demo implementation

cobras

`https://dtai.cs.kuleuven.be/software/cobras/`