Feature-driven Time Series Clustering

Donato Tiano
Univ. Lyon 1

Angela Bonifati
Univ. Lyon 1

Raymond Ng
Univ. British Columbia

Hi, my name is Donato Tiano, I am a PhD student at Lyon 1 University. Today I will present my paper entitled Feature-Driven Time Series Clustering (joint work with Angela Bonifati and Raymond Ng).

Introduction

We present FeatTS, a Semi-Supervised Clustering method that leverages features extracted from the raw time series to create clusters that reflect, as much as possible, the original time series.

To the best of our knowledge, FeatTS is the first feature-based semi-supervised clustering framework with these key properties.

FeatTS semi-supervised algorithm that uses the features extracted from the time series in order to cluster them. To the best of our knowledge, FeatTS is the first feature-based semi-supervised clustering framework with these key properties.

Related Work

The FeatTS algorithm leverages the concepts of Constrained Clustering, more specifically Clustering by Seeding. This method uses a small amount of labels of the original dataset in order to create two kinds of links, i.e. Must Link and Cannot Link.

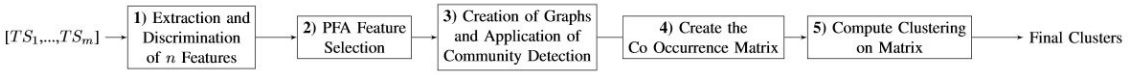
1. Must Links are connections between two data points that represent a “constraint of belonging''. This means that the data points (or time series at large) should be clustered together.
2. Cannot Links are connections that represent a “non-belonging constraint” thus leading to separate data points.

Seeded kMeans[1] is the most representative method in this category

[1]Sugato Basu, Arindam Banerjee, and Raymond Mooney. 2002. Semi-supervised clustering by seeding. In Proceedings of ICML.

FeatTS is based on Clustering by Seeding concept. The idea is to create two kinds of connections between the data of the dataset, called Must Link and Cannot Link.
With the Must Link we are defining a "Constraint of belonging" between the two data. It means that two data points should be clustered together.
By opposite, the Cannot Link defines a "Non Belonging Constraint", hence going to separate the data points.
Seeded KMeans is the most prominent algorithm of this category.

FeatTS: the algorithmic pipeline

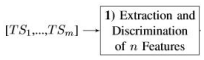


In this figure we can see the 5 main steps of our algorithm.

Extraction and Selection Features

The first step of the algorithm is the extraction, through a library called TSFresh, of all the possible features derived from the time series.
This operation allows to extract several hundreds of features from an input dataset. Therefore, it becomes pivotal to discriminate the most significant ones.

Time Series	*mean*	*trend_stderr*	*peaks*	*quantile*	*trend_rvalue*	Length	...	Label
TS_1	51.3	3.51	8	57	-0.94	89	...	No Kidney Failure
TS_2	40.6	4	5	43	-0.55	206	...	~~No Kidney Failure~~
TS_3	74.3	17	10	106	0.01	159	...	Kidney Failure
TS_4	95.8	9.4	10	85	0.43	139	...	~~Kidney Failure~~

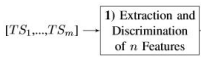


The first step is the extraction of the features. Indeed, using a library called TSFresh, we can extract numerous features. Indeed, as indicated in the figure, we can extract plenty of features completely different. Among all the features extracted, only some of them are really useful in order to obtain good performance. Therefore, we need a method that permits to discriminate which features have the best correlation with the dataset.

Extraction and Selection Features

The feature selection process leverages the supervisioned procedure of Benjamini-Yekutieli which takes advantage of a small subset to obtain the p-value of each feature and thus its discriminating value.

Feature	p-value
quantile	0.006
trend_stderr	0.008
trend_rvalue	0.010
peaks	0.09
Length	0.20
mean	0.45
...	...



The operation of discrimination of the features is done via a procedure called Benjamin and Yacutelli Procedure. This procedure needs a small subset of Supervised Data, in order to compute the p-value for each feature useful for considering the importance of each of them.
In the figure, we can see how the quantile feature is the most discriminative according to the Benjamin & Yakutelli Procedure.
However, this procedure doesn't let isolate the number of features useful for clustering.

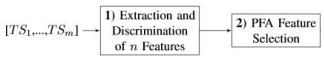
PFA Features Selection

We apply a technique called Principal Feature Analysis (PFA). PFA preserves the original values of the features and thus the distance between them.
We can leverage the concept of explained variance, representing the ratio between the variance of one single feature and the sum of variances of all individual features.

Feature	p-value	Expl. Var.	Sum Expl. Var.
quantile	0.006	0.45	0.45
trend_stderr	0.008	0.35	0.80
trend_rvalue	0.010	0.10	0.90
peaks	0.09	0.04	0.94
Length	0.20	0.01	0.95
mean	0.45	0.09	0.959
...	...	...	...

90% E.V.

quantile
trend_stderr
trend_rvalue

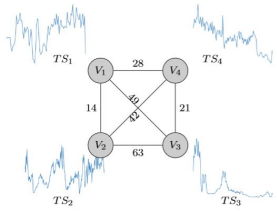


Therefore, in order to obtain the optimal subset of features, FeatTS adopts an algorithm called Principal Feature Analysis. PFA computes the explained variance of each feature and, based on a set threshold, selects the minimum number of features that need to obtain the threshold requested. In the figure, we can see that fixing a threshold equal to 90% of the explained variance, this can already be obtained with the first three features of the left-hand table. Therefore, only the first 3 features will be used by the algorithm from now on.

Creation of Graphs and Application of Community Detection

For each feature chosen by PFA, FeatTS creates a different edge-weighted graph network where the nodes represent the time series of the initial dataset and the edge-weighted are computed by subtracting, in absolute value, the value of the feature of two connected time series.

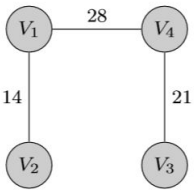
Time Series	quantile
TS_1	57
TS_2	43
TS_3	106
TS_4	85



Once we select the features, we convert the time series and their relationships into edge-weighted graphs. The encoding of time series into edge-weighted graphs allows us to capture the global relationships among the raw time series samples. Therefore, we convert each time series in an edge-weighted graph where the vertices represent the time series while the edges represent the distance between the connected nodes of the edge, i.e. the difference between the values of the feature of two different time series.
In the figure we can see that the weight between V1 and V2 is 14 because of the difference between the quantile features of Time Series 1 and Time Series 2.

Creation of Graphs and Application of Community Detection

FeatTS orders all the distances computed in an ordered list and then requires to the user to choose the percentage of distances that he wants to keep starting from the lowest distances from the previously computed list.



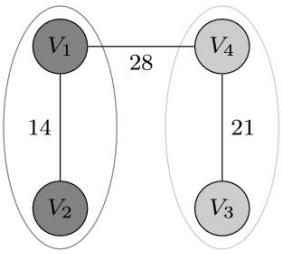
V_1	V_2	**14**
V_3	V_4	**21**
V_1	V_4	**28**
V_2	V_3	42
V_1	V_3	49
V_2	V_4	63

Threshold: 50%

In order to capture the similarity, we only retain in the graphs the edges whose weight are less than a given threshold distance. Therefore, once ordered in ascending way all the distances and select a threshold, FeatTS cuts all the edges with the distances higher than the maximum fixed. In the figure we decide to keep the 50% of edges, so all the edges with a distance higher than 28 will be removed from the graph.

Creation of Graphs and Application of Community Detection

Finally, FeatTS applies a community detection algorithm in order to search for groups of densely connected vertices forming communities.



Finally, once pruned the graph we will apply an algorithm of Community Detection in order to extract the communities from the graph. Feats apply this routine on all the graphs obtained by the features. Therefore, we need a structure to combine all the results obtained in one single data structure.

Creation of Co-Occurrence Matrix

Intuitively, the more times the time series are placed within the same community, the more similar they are. FeatTS creates a matrix in which the rows and columns contain all the time series of the dataset.
Each cell x_{ij} in the matrix corresponds to the similarity between time series T_i (in row i of the matrix) and T_j (in column j of the matrix).

Dataset	TS_1	TS_2	TS_3	TS_4
TS_1	x_{11}	x_{12}	x_{13}	x_{14}
TS_2	x_{21}	...	...	...
TS_3	x_{31}	...	...	...
TS_4	x_{41}	...	x_{ij}	...

FeatTS uses a structure called Co-Occurrence matrix to combine all the communities in one single structure. The Co-Occurrence matrix is a matrix where the columns and rows represent all the time series of the dataset, and each cell represents a similarity value between these time series.
The similarity of two time series increases with the number of times the two time series are clustered together.

Creation of Co-Occurrence Matrix

FeatTS allows to assign a weight to the features based on the number of communities the features have generated. The weight is higher for features with a number of communities equal to the number of required clusters.
Let w_i be a weighting function defined on feature F_i, C the number of clusters requested by the user and O_i the number of communities of the feature F_i

$$\begin{cases} w_i = \frac{C}{O_i}, \ if \ O_i > C \\ w_i = \frac{O_i}{C}, \ if \ C > O_i \\ w_i = 1, \ otherwise \end{cases}$$

In order to make more robust the computation of the similarity, each feature has weights that depend by the number of communities found by the Community Detection Algorithm. The weight will be computed dividing the number of communities of the features by the number of requested clusters if the number of communities is higher than the number of clusters. Vice versa, if the number of requested clusters is higher than the number of communities extracted, we invert the numerator and denominator of the function. The weight is equal to 1 in case the two values are equal.

Creation of Co-Occurrence Matrix

$$quantile = \{(TS_1, TS_2), (TS_3, TS_4)\}$$
$$trend\_stderr = \{(TS_1), (TS_2), (TS_3, TS_4)\}$$
$$trend\_rvalue = \{(TS_1, TS_2, TS_3, TS_4)\}$$

2 clusters requested →

Weights

quantile : 1
trend_stderr : ⅔
trend_rvalue: ½

Co-Occurrence Matrix

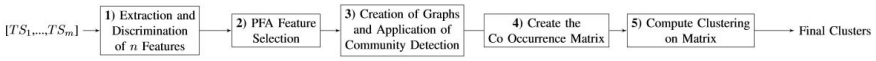
Dataset	TS_1	TS_2	TS_3	TS_4
TS_1	1	$\frac{1+0.5}{0.66+1+0.5}$	$\frac{0.5}{0.66+1+0.5}$	$\frac{0.5}{0.66+1+0.5}$
TS_2	$\frac{1+0.5}{0.66+1+0.5}$	1	$\frac{0.5}{0.66+1+0.5}$	$\frac{0.5}{0.66+1+0.5}$
TS_3	$\frac{0.5}{0.66+1+0.5}$	$\frac{0.5}{0.66+1+0.5}$	1	$\frac{0.66+1+0.5}{0.66+1+0.5}$
TS_4	$\frac{0.5}{0.66+1+0.5}$	$\frac{0.5}{0.66+1+0.5}$	$\frac{0.66+1+0.5}{0.66+1+0.5}$	1

As an example, here we show the weights of the three chosen features corresponding to the number of communities found in each graph encoding of the respective features. Indeed, the algorithm for community detection finds for the trend_StdError 3 different communities. In this case we have divided 2 by the number of the extracted communities, i.e. 3. In the other feature, called trend_rvalue, only one community is found thus we divide 1 by 2.

Clustering

We need one more intermediate step, i.e. to compute the distances between the rows of the Co-Occurrence Matrix. We employ a standard Euclidean distance to perform the row comparison.
Finally, we apply the standard K-Medoid algorithm on the distances computed above. K-Medoid allows us to extract clusters of time series that have the smallest distance among them.

Dataset	TS_1	TS_2	TS_3	TS_4
TS_1	1	0.69	0.23	0.23
TS_2	0.69	1	0.23	0.23
TS_3	0.23	0.23	1	1
TS_4	0.23	0.23	1	1



Once we have computed the co-occurrence matrix, we can apply the kMedoid clustering algorithm on the matrix in order to extract the corresponding clusters.

Results

We used 64 benchmark datasets belonging to the UCR collection[1], including both real-life and synthetic datasets. The entire list of datasets used for benchmark and code for the reproducibility is available online[2]. The results are expressed in Adjusted Mutual Information(**AMI**)

Dataset	FeatTS	SeededKMeans
Adiac	0,31	**0,52**
MoteStrain	**0,48**	0,02
TwoLeadECG	**0,88**	0,07
ECG200	**0,34**	0,06
Computers	0,09	0,01
Coffee	**1**	0,88
GunPoint	**0,52**	0
Arrowhead	**0,29**	0,27
ItalyPowerDemand	**0,54**	0
Meat	0,4	**0,75**
OliveOil	0,27	**0,53**
Trace	**0,74**	0,69
Wine	**0,12**	0,01
Worms	**0,16**	0,12
ShapesAll	0,08	0,45

[1] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. 2018. The UCR Time Series Archive.
[2] https://github.com/DonaTProject/FeatTS

In order to evaluate the performance of our algorithm,
we first used the datasets of the UCR collection.
This collection consists of about 64 datasets, in the figure we show an excerpt of the results obtained on the datasets. We can see how the results obtained by our algorithm, which are expressed through the adjusted mutual information, exceed those of Seeded kMeans (used as a baseline).

Results

We use real-life medical time series courtesy of the Personalized Medicine Department at the European Hospital George Pompidou in Paris. These time series contain signals from patients suffering from kidney diseases.
We ran experiments on two variants of this dataset. The first variant named $Kidney3Yr$ contains 222 patients (one time series per patient) and spans 1 to 3 years with a variable length between 90 and 230 data points in the time series. The second variant called $Kidney5Yr$ is composed of 278 patients spanning 5 years with time series having roughly 100 data points.

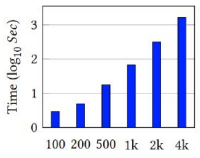
The results are expressed in Adjusted Mutual Information(**AMI**)

Dataset	FeatTS	SeededKMeans
Kidney3Yr	**0.56**	0.44
Kidney5Yr	**0.58**	0.48

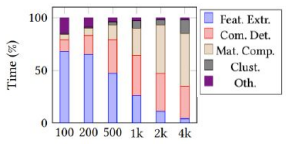
The second dataset is a real dataset that was provided to us by the George Pompidou hospital in Paris. The latter consists of patients who suffer from kidney disease and for whom we need to predict whether they need dialysis or not.
We have two variants of dataset, whose difference lies in the number of patients per dataset and the length of the measurement.
We can see how the results obtained by our algorithm outperform those of Seeded kMeans.

Results

We have also assessed the scalability of our method by increasing both the number and length of time series in a dataset. In this experiment, we have used synthetic time series generated with GRATIS allowing for a controlled generation of time series by using diverse characteristics as spectral entropy, trend, seasonality, stability, etc.

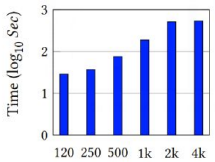


(a) Time vs. dataset size

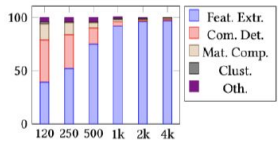


(c) % Time of each component vs. dataset size.

Increasing Dataset



(b) Time vs. TS length



(d) % Time of each component vs TS length.

Increasing Length

Finally, we tested the scalability in terms of time of FeatTS increasing the size of the dataset with the fixed length of the time series and increasing the length of the time series but leaving the same number of time series.
We can see how if we increase the number of time series, the time, expressed in log scale, clearly increases. Moreover, we can see that the majority of the time is spent for the creation of the co-occurrence matrix.
The situation is somehow opposite when we increase the length of the time series. In that case most of the time is spent to extract the features from the time series.

Conclusion

- Our work on clustering of time series shows that there is no one-size-fits-all solution regarding the set of features to use. In fact, we leveraged the features drawn from the data itself rather than taking a predefined set of features for all the datasets. Our flexible graph encoding allows us to process the most significant features in parallel and the further steps of our method allow us to combine the results.

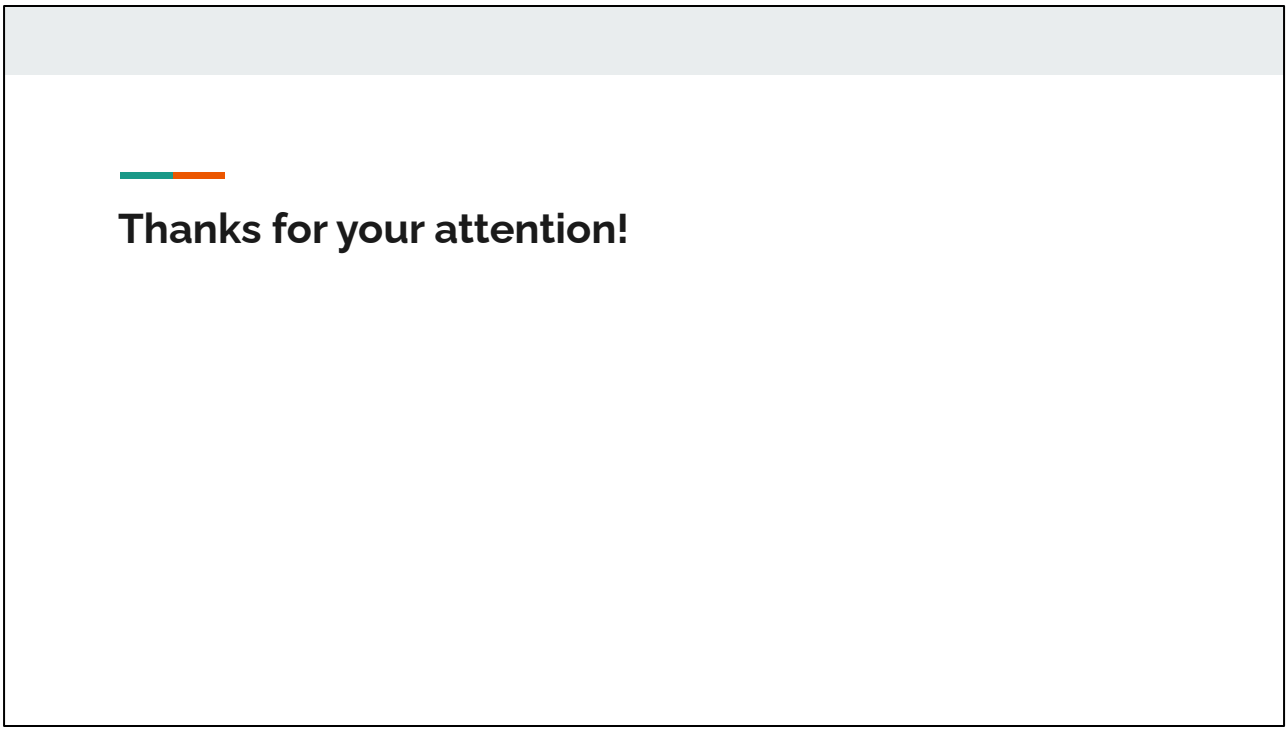
- Upcoming: FeatTS will be demonstrated at SIGMOD 2021, please stay tuned!

We have presented FeatTS, a system capable of doing time series clustering while leveraging custom features for each dataset that best fit the dataset itself. This overcomes the results in state of the art algorithms relying on raw data or on a fixed set of features.
FeatTS will be demonstrated at Sigmod 2021 so please stay tuned!

Future work

This work could be improved by rendering the entire pipeline unsupervised instead of the current semi-supervised approach.
Another improvement would be to dynamically choose the threshold for graph creation based on the processed features.
Finally, the weights of the community detection algorithm could be combined with relevance degrees of the features.

In the next future we plan to make the pipeline completely unsupervised and we also want to be able to set a dynamic threshold to be used in the graph encoding.
Finally, we are evaluating if the use of relevance weights extracted from the supervised procedure can be used in combination with the current weights.



Thanks a lot for your attention!