HAL
open science

FeatTS: Feature-based Time Series Clustering

Donato Tiano, Angela Bonifati, Raymond Ng

To cite this version:

Donato Tiano, Angela Bonifati, Raymond Ng. FeatTS: Feature-based Time Series Clustering. SIGMOD/PODS '21: International Conference on Management of Data, Jun 2021, Virtual Event, China. pp.2784-2788, 10.1145/3448016.3452757 . hal-03548278

HAL Id: hal-03548278

https://hal.science/hal-03548278v1

Submitted on 9 Feb 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

FeatTS: Feature-based Time Series Clustering

Donato Tiano
donato.tiano@univ-lyon1.fr
Lyon 1 University
France

Angela Bonifati
angela.bonifati@univ-lyon1.fr
Lyon 1 University
France

Raymond Ng
rng@cs.ubc.ca
University of British Columbia
Canada

ABSTRACT

Clustering time series is a recurrent problem in real-life applications involving data science and data analytics pipelines. Existing time series clustering algorithms are ineffective for feature-rich real-world time series since they only compare the time series based on raw data or use a fixed set of features for determining the similarity. In this paper, we showcase FeatTS, a feature-based semi-supervised clustering framework addressing the above issues for variable-length and heterogeneous time series. Specifically, FeatTS leverages a graph encoding of the time series that is obtained by considering a high number of significant extracted features. It then employs community detection and builds upon a Co-Occurrence matrix in order to unify all the best clustering results. We let the user explore the various steps of FeatTS by visualizing the initial data, its graph encoding and its division into communities along with the obtained clusters. We show how the user can interact with the process for the choice of the features and for varying the percentage of input labels and the various parameters. In view of its characteristics, FeatTS outperforms the state of the art clustering methods and is the first to be able to digest domain-specific time series such as healthcare time series, while still being robust and scalable.

CCS CONCEPTS

• **Computing methodologies** → *Feature selection*; **Cluster analysis**; • **Mathematics of computing** → *Time series analysis.*

KEYWORDS

Semi-supervised Clustering; Community Detection; Features Selection; Clustering for Data Science.

ACM Reference Format:
Donato Tiano, Angela Bonifati, and Raymond Ng. 2021. FeatTS: Feature-based Time Series Clustering. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21), June 20–25, 2021, Virtual Event , China.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3448016.3452757

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. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGMOD '21, June 20–25, 2021, Virtual Event , China
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8343-1/21/06…$15.00
https://doi.org/10.1145/3448016.3452757

1 INTRODUCTION

In this paper, 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. Our approach differs from existing methods in the literature as it employs the features of the time series while existing methods focus on the similarity of the time series themselves [8]. The FeatTS algorithm leverages the concepts of Constrained Clustering, more specifically Clustering by Seeding. Seeded kMeans [1] is the most representative method in this category, as it 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. 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. Cannot Links are connections that represent a "non-belonging constraint" thus leading to separate data points. Leveraging these two kinds of constraints, Seeded kMeans allows to discover clusters that respect them.

The novelty of FeatTS consists in automatically selecting the most appropriate statistical features based on the dataset provided as input. In fact, not all the features have the same quality and choosing a subset of high-quality features for each dataset is beneficial for the clustering step. This characteristic of FeatTS turns to be fruitful in several real-life data science and data analytics pipelines. Indeed, the features of time series are interpretable by humans, thus leading to a more transparent and human-centric clustering process. To the best of our knowledge, FeatTS is the first feature-based semi-supervised clustering framework with these key properties.

In our demonstration, we show how FeatTS works on domain-specific time series (among the others), by focusing on the time series of patients suffering from end-stage kidney diseases. These time series encode the variations over time of the Glomerular Filtration Rate (GFR) signal, estimating how much blood passes through the glomeruli each minute. How GFR changes is crucial for the patient's survival, since rapidly descending values of GFR over time indicate a dangerous condition that might lead to kidney failure and even death. A more stable evolution of GFR over time is less worrisome for the concerned patient, and might lead to guide appropriate treatment. The hard question here is how to select the subset of features that help medical doctors discriminate the patients based on the label while performing clustering.

Once the features are selected, FeatTS computes the global relationships between the time series based on their statistical features. It uses graph networks to obtain such an encoding. Indeed, FeatTS converts each time series into nodes and creates weighted edges between such nodes. Each edge represents the distance between the connected nodes, i.e. the difference between the values of two different time series using the selected feature. FeatTS prunes the

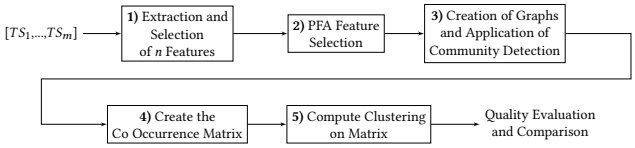

Figure 1: The algorithmic pipeline of FeatTS.

graphs based on a threshold and applies a Community Detection algorithm in order to obtain the global relationship among time series. As a final step, FeatTS will holistically merge the results of the communities into a Co-Occurrence matrix, on which a clustering algorithm (K-Medoid) is applied. Further details about the method can be found in [7]. In this demo, we showcase all the steps of the FeatTS pipeline by letting the user transparently interact with the clustering process and visualize the intermediate results of the clustering. To the best of our knowledge, FeatTS is the first system to exhibit the following characteristics:

- It builds upon a novel semi-supervised clustering method leveraging global relationships among the most discriminating features extracted from the time series.
- As opposed to previous work, it does not consider the similarity of the raw data and it only focuses on the features of the time series. As such, it allows to treat at par all the features of a given dataset instead of pre-selecting a fixed number of features for all possible datasets.
- FeatTS is shown at work on a real-life healthcare dataset, thus leading to cluster patients based on their risk assessment.
- FeatTS has proved to outperform previous approaches on varied-length time series, while still keeping satisfactory robustness and scalability.

2 SYSTEM OVERVIEW

In this section, we detail the main steps of the FeatTS pipeline as exemplified in Figure 1.

2.1 Features Extraction and Selection

The first phase, corresponding to Step 1) in Figure 1 is the extraction of the features from the time series. We consider feature extraction methods available in the literature and in readily predefined libraries [2].

The TSfresh[2] library allows us to extract a large number of features. However, feature selection becomes pivotal, since not all the features have the same relevance for the clustering steps.

The selection of the most discriminating features will be done through the use of time series labels that will be provided as input on a small subset of the time series. We will therefore use a supervised procedure called Benjamini-Yekutieli where its otuput will be a list of features ranked by their p-values.

Once the order of importance of each feature has been obtained, it is necessary to choose which of them can be used to create the graphs encodings. Among the numerous algorithms for feature selection, we decided to opt for PFA [5], a variation algorithm of Principal Component Analysis (PCA).The key difference is that PFA preserves the original values of the features and thus the distance between them. Therefore, 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. We set a threshold t of the explained variance equal to 0.9. It means that, once ordered the features using the Benjamini-Yekutieli procedure, FeatTS chooses the minimum number of features for which the sum of their explained variance reaches the 90% of the variance produced by the remaining features.

2.2 Graph Rendering and Community Detection

We convert the time series and their relationships into edge-weighted graphs. The encoding of time series into edge-weighted graphs allows us to represent our clustering problem in another dimension space without loss of information and becomes crucial in order to capture the global relationships among the raw time series samples.

For each feature F_i chosen by PFA in the previous step, we create a different edge-weighted graph network where the nodes represent the time series of the initial dataset. The nodes are connected via weighted edges, where the weights are computed by subtracting the absolute values of the feature F_i of two connected time series.

After computing all these distances and creating a fully connected graph network for each feature, we desire to keep only the most important links between the nodes. Therefore, we need to prune the graph adopting a straightforward technique. At the beginning, we rank all the distances computed for the features F_i in a ranked list and then we ask the user to choose the percentage of distances to be kept starting from the lowest distances from the previously computed list. Hence, the distances that are discarded will represent the portion of the graph that has been pruned.

Once we have obtained one graph for each feature, we want to cluster the vertices of the graphs that represent the time series of the dataset. The techniques employed for clustering vertices in a graphs are community detection (CD) algorithms. These algorithms search for groups of densely connected vertices forming communities. Among the different tested algorithms, we have opted for the Greedy Modularity Algorithm [6]. This algorithm turns out to strike a balance between speed and robustness and does not require any additional input parameter other than the graphs themselves. The communities extracted on each graph-feature varies from one graph to another graph. Therefore, we need a method for unifying the different communities in order to obtain understandable results.

2.3 Creation of the Co-Occurrence Matrix

The underlying intuition is that if two time series are similar, they will be similar for the majority of their discriminating features. We employ a *Co-Occurrence matrix* to put this in practice. The matrix consists of recording for each pair of time series how many times they are grouped within the same community. Intuitively, the more times they are placed within the same community, the more similar the time series are.

Thus, we create 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). A naive solution is to compute the value of the cell x_{ij} counting the number of times that

the two time series TS_i and TS_j fall within the same community divided by the number of features. In this way, the similarity grows as the number of times the time series that are found in the same community grows. However, the assignment to a community does not take into account the quality of the features of the underlying time series. Therefore, the similarity between the time series shall leverage the quality of each feature. The goal is to prioritize the features for which the number of communities is sufficiently close to the number of clusters requested by the user. Hence, the similarity between two time series will be stronger in the feature with the same number of communities requested by the user. To achieve that, we assign an approximate weight to each feature, based on the number of communities that the Community Detection algorithm computes. Let w_i be a weighting function defined on each feature F_i as follows:

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

where C is the number of clusters expected by the user and O_i is the number of communities extracted by the CD algorithm on the feature F_i. Hence, the weights will be higher if the number of obtaining clusters O is equal or sufficiently close to C and lower otherwise. Not surprisingly, instead of simply counting the number of times that the time series TS_i and TS_j co-occur in the same cluster, we now sum the weights of the features where they are detected together and divide by the sum of the weights of all the features. This gives us more meaningful weights than the former previously considered.

2.4 Clustering Time Series

With the Co-Occurrence matrix computed, we can quantify the similarity between two time series. In order to prepare the creation of the time series clusters, we need to calculate the distances between the rows of the Co-Occurrence Matrix adopting the Euclidean distance. Finally, we apply the standard K-Medoid algorithm [4] on the distances computed above. K-Medoid allows us to extract clusters of time series that have the smallest distance among them.

3 DEMONSTRATION ROADMAP

Our demonstration of FeatTS aims to highlight the key aspects of the system. In particular, we focus on the capability of FeatTS to clusterize the time series with the number of clusters provided in input by the user, to modify the threshold of each feature in order change the detected communities as well as to handle the increase of the number of time series in the dataset and the increase of their corresponding lengths.

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. A global assessment of renal function is often ascertained by estimating the rate of filtration, called the glomerular filtration rate (GFR). GFR estimates how much blood passes through the glomeruli each minute. Kidney failure occurs when GFR is under $90mL/min/1.73m^2$, whereas when it drops to $15mL/min/1.73m^2$, it means that the patient needs dialysis or a transplant. Since these are

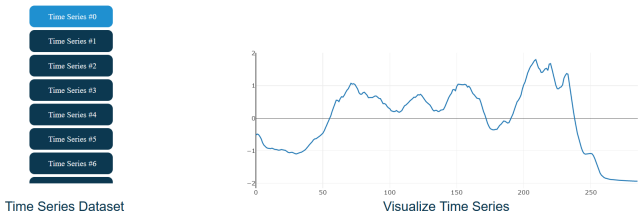

Figure 2: Visualization of the Time Series

invasive operations, it is important to understand if a sudden drop in the GFR occurs. In this case, medical doctors might recommend urgent surgery or to resort to dialysis depending on the GFR values over time. The time series provided by the Hospital exhibit two classes, namely "Kidney Failure" and "Not Kidney Failure". The goal is to detect the time series that do not have labels and to visualize the differences between the two classes.

Despite we center the discussion around the GFR dataset, we will showcase our system on other datasets from the literature, i.e. the UCR Time Series Archive [3]. Furthermore, in order to show the scalability of the system, we will also use two synthetic dataset. The first one is composed by increasing incrementally the number of time series in the dataset (100, 200, 500, 1k, 2k, 4k) but fixing the length (60 points). In the other dataset, we have incrementally augmented the number of points in the time series (120,250,500,1k,2k,4k) but fixing the number of time series (500).

3.1 Executing the End-to-end Pipeline

For ease of exposition, in the following we discuss the features of FeatTS on the aforementioned GFR Dataset.

At the very beginning, the system will allow to set up the parameters for initiating the clustering process. In the first sliding bar ("Cutting Threshold"), we set the percentage of distances that we want to keep for each ordered list of distances created for each chosen features. In this scenario, we decided to keep the 80% of ordered distances for each feature. Usually, keeping the 80% of the distances provides good results, but this value can be modified for each feature consequently. The second sliding bar ("Learning Threshold") permits to select a subset of time series labels in order to obtain the best features for dataset. In our scenario, we used the 20% of the labels provided by the Kidney dataset, which let us obtain good results with a low number of labels. This value strongly depends on the degree of supervision that the user is willing to rely on. Indeed, the higher the threshold is, the more labels FeatTS will use and the higher is the quality of the features and of the clustering result. The percentage of labels can be also set to 0, thus making the approach completely unsupervised.

Finally, FeatTS makes possible to choose the number of clusters to extract from a given dataset regardless of the classes within the labels. This possibility is interesting for extracting new information from the dataset. In thi scenario, we set to 2 the number of clusters that we want to obtain [1] and choose the Kidney Dataset.

Figure 2 shows a board visualizing the time series of the datasets. The user can select the time series that he/she wants to visualize and cumulates them in the same plot by using the left menu.

[1]Notice that this number can be arbitrarily different from the initial number of classes provided with the dataset.

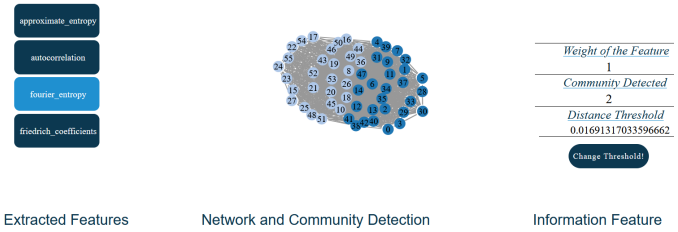

Figure 3: Networks and Features Visualization

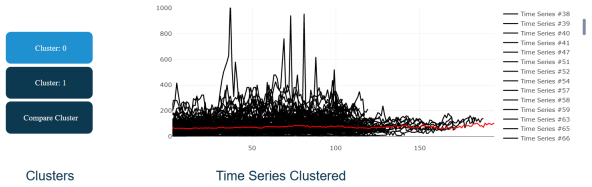

Figure 4: Clustered Time Series

Figure 3 shows the feature selection and graph encoding steps. Precise information about the features such as their weights within the Co-Occurrence Matrix, the number of communities found and the distance threshold are visualized for each feature corresponding to the showed graph encoding. The information shown in Figure 3 is important as the user can understand the number of communities detected by the Greedy Modularity algorithm for a specific feature. As will be shown in the second scenario, FeatTS offers the opportunity to modify the cutting threshold of a specific feature, thus allowing the user to customize the graph and the number of communities found.

Finally, Figure 4 shows a screenshot of the system where the obtained time series in a given cluster are visualized. This screenshot is useful if the user wants to visualize some specific time series that belong to a single cluster. The ensemble of the time series of that cluster are shown in black and can be selected from the legend on the right while the median of all the time series of that cluster is highlighted in red. This median time series is further used by the system by enabling a comparison across clusters as shown in Figure 5. The computation of the median of the time series is an excellent way to visualize the average trend of the time series of a given cluster. Moreover, by comparing the medians of each cluster, it can also be used to assess the difference in trends between the various clusters.

As an example, we could appreciate the remarkable difference between the medians of cluster 0 and cluster 1. The former cluster represents the patients who have suffered from a sudden drop of GFR and who needed an invasive a treatment as opposed to patients in cluster 1, who also suffered from a significant drop of GFR but did not need invasive treatment.

3.2 Personalizing the Distance Threshold

In this scenario, we will illustrate how the user can customize the threshold cutting of each feature in FeatTS. Indeed, after the selection of the initial threshold, Figure 3 shows the board in which the user can select one of the features extracted by the system and, using the "Change Threshold" button, modify the threshold. The

Figure 5: Compare Median Cluster Time Series

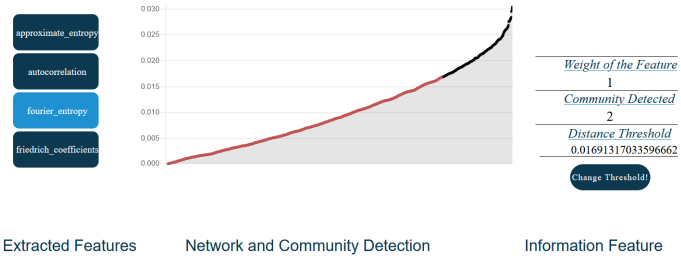

Figure 6: Change Threshold

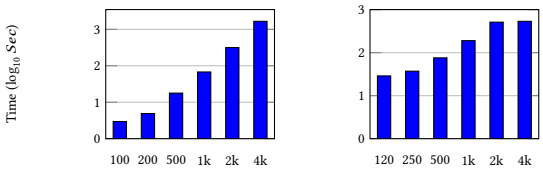

Figure 7: Runtimes with increasing dataset size (lhs) and time series lengths (rhs).

system will show as output the view depicted in Figure 6 showing the plot of the distances computed for each feature (on the left part of the screenshot). The points in the plot are divided into two halves: the first half (in red) represents the distances included in the initial view, while the second half (in black) are the excluded distances. The system allows to vary the balance between included and excluded distances, thus bringing to regenerate the graph encoding (as illustrated in Figure 3).

3.3 System Scalability

In the third and last scenario, we will showcase the scalability and robustness of FeatTS when increasing the number of time series and their lengths using the synthetic datasets [2]. Figure 7 shows (in logarithmic scale) how the runtime grows very quickly when we increase the number of time series within the dataset (lhs). The increase is milder in the case of variation of the length of the time series (rhs). These runtimes can be explained by taking into account the distance computation for all the features and the creation of the Co-Occurrence matrix, that heavily depends on the number of time series.

4 ACKNOWLEDGEMENTS

Research funded by ANR (grant nr. 18-CE23-0002 QualiHealth).

[2] A notebook on this scenario can be be accessed at https://shorturl.at/noCDJ

REFERENCES

[1] Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. 2002. Semi-supervised Clustering by Seeding. In *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002), University of New South Wales, Sydney, Australia, July 8-12, 2002*, Claude Sammut and Achim G. Hoffmann (Eds.). Morgan Kaufmann, 27–34.

[2] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. 2018. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh - A Python package). *Neurocomputing* 307 (2018), 72–77. https://doi.org/10.1016/j.neucom.2018.03.067

[3] Hoang Anh Dau, Anthony J. Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn J. Keogh. 2019. The UCR time series archive. *IEEE CAA J. Autom. Sinica* 6, 6 (2019), 1293–1305. https://doi.org/10.1109/jas.2019.1911747

[4] Anil K. Jain and Richard C. Dubes. 1988. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[5] Yijuan Lu, Ira Cohen, Xiang Sean Zhou, and Qi Tian. 2007. Feature selection using principal feature analysis. In *Proceedings of the 15th International Conference on Multimedia 2007, Augsburg, Germany, September 24-29, 2007*, Rainer Lienhart, Anand R. Prasad, Alan Hanjalic, Sunghyun Choi, Brian P. Bailey, and Nicu Sebe (Eds.). ACM, 301–304. https://doi.org/10.1145/1291233.1291297

[6] Mark E. J. Newman. 2010. *Networks: An Introduction*. Oxford University Press. https://doi.org/10.1093/ACPROF:OSO/9780199206650.001.0001

[7] Donato Tiano, Angela Bonifati, and Raymond Ng. 2021. Feature-driven Time Series Clustering. In *Proceedings of EDBT*.

[8] Haishuai Wang, Qin Zhang, Jia Wu, Shirui Pan, and Yixin Chen. 2019. Time series feature learning with labeled and unlabeled data. *Pattern Recognit.* 89 (2019), 55–66. https://doi.org/10.1016/j.patcog.2018.12.026