Human-Centered Clustering for Time Series Data

Donato Tiano, Angela Bonifati, Raymond Ng

Human-Centered Clustering for Time Series Data

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

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

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

ABSTRACT

Clustering is a fundamental step of several data science pipelines leading to compute groups of objects with high similarity. In this paper, we focus on the problem of clustering time series and we show how we can make this process human-centered. In particular, by relying on FeatTS, a feature-based semi-supervised framework, we show how parameters can be tuned by the users in an intuitive fashion. The considered parameters are: (1) the learning threshold leading to choose how much labeled data are needed as input; (2) the cutting threshold leading to prune time series whose distance is below this threshold; (3) the number of clusters desired as output leading to create different clusters than the number of classes reflected by the labeled data. In particular, while the first two parameters allow to tune the amount of the supervision and tweak the number of significant features, the third parameter leads to showcase the robustness of the clustering process with respect to the number of clusters provided as input. Finally, the significance of the entire multi-step clustering pipeline is empirically demonstrated through a handful of ablation tests.

ACM Reference Format:
Tiano Donato, Angela Bonifati, and Raymond Ng. 2021. Human-Centered Clustering for Time Series Data. In *DaSH@KDD'21, August 15-16, 2021, Virtual Conference.*. ACM, New York, NY, USA, 7 pages.

1 INTRODUCTION

Clustering time series is a crucial task in data-intensive pipelines. Nowadays, several algorithms exist, ranging from unsupervised to semi-supervised [1, 5]. User interaction also varies from one algorithm to another. For instance, active semi-supervised algorithms [11] ask the user if two time series are equal throughout the entire process of cluster creation. Seeded kMeans [2] is a semi-supervised clustering by seeding that asks the user to provide in input the labels of a portion of the dataset, used for creating the final cluster. In this case, the user provides an input at the beginning of the process and not along the entire procedure. On the other hand, other approaches [6] are completely unsupervised. Therefore, the user cannot participate in the process of cluster creation at all.

Prior work has shown the utility of exploiting features in semi-supervised clustering for variable-length time series [9, 10]. In this paper, we leverage this work [9, 10] by describing, probing and assessing the human-centric aspects of the clustering pipeline by allowing the user to exploit a novel feature-based viewpoint of the process. We show how to tune the entire algorithmic pipeline of the algorithm, called FeatTS, a human-centered semi-supervised clustering method that leverages features extracted from the raw time series to create clusters that reflect, as much as possible, the user's requirements.

This 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 [12]. This approach is superior to other semi-supervised approaches (such as Seeded K-Means [2]) and unsupervised ones (such as K-Shape [6]), as shown in [10].

The novelty of FeatTS consists in automatically selecting the most appropriate statistical features based on the dataset and the user's requirements 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 out 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.

FeatTS allows to select the most appropriate statistical features based on a parameter called 'Learning Threshold'. This parameter allows to tune the amount of supervising for obtaining the features.

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 graphs based on a parameter called 'Cutting Threshold'. This parameter allows to choose how many edges should be kept between the nodes. Therefore, the Cutting Threshold allows to tune the similarity between the time series in the final cluster. Once obtained the pruned graphs, FeatTS applies a Community Detection algorithm in order to obtain the global relationship between time series.

In the final step, FeatTS permits to choose the number of final clusters to obtain. Indeed, by tuning the parameter called 'Number of Clusters', the user can obtain a different number of clusters from the classes provided in the supervised dataset. Therefore, FeatTS will holistically merge the results of the communities in a Co-Occurrence matrix, on which a clustering algorithm (K-Medoid) is applied. Further details about the method can be found in[10].

In the next sections, we show how FeatTS interacts with the user for creating clusters corresponding to the user's needs via the three fundamental parameters explained in Section 3. Moreover, we show the impact of the parameters on the creation of the cluster and their

flexible tuning. In Section 4, we describe the experimental setup on real-life data science workflows and on benchmarking datasets. Section 5 presents the importance of each step of the FeatTS pipeline by means of a handful of ablation tests. Indeed, we show the results obtained by FeatTS when some steps of the pipeline are replaced with an alternative step. We show how the quality of the obtained clusters deteriorates, thus proving the effectiveness of the clustering pipeline. Finally, Section 6 concludes the work and pinpoints future directions of investigation.

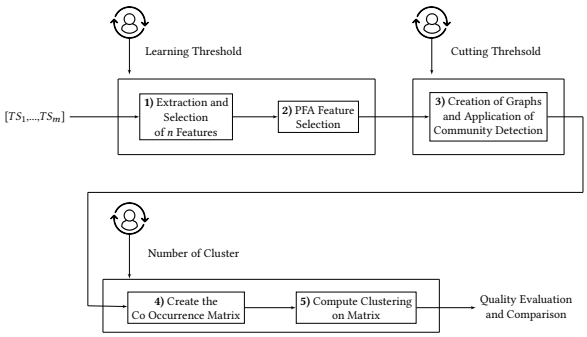


Figure 1: The algorithmic pipeline of FeatTS.

2 RELATED WORK

Semi-supervised learning is a combination of supervised and unsupervised learning. It uses limited information provided by the user and a large amount of unknown data in order to train a model. A subcategory of semi-supervised clustering is Constrained Clustering, which relies on a user to create two kinds of constraints, i.e. Must Link and Cannot Link. The Must Link connection between two data points means that the data points (or time series at large) should be clustered together. Cannot links do the opposite, thus leading to separate data points.

There exist various methods to create these constraints which can be divided in two main categories: Active Learning and Clustering by Seeding. The Active Learning [13] algorithms allow users to interact during the entire process of creation of the cluster for modifying the various clusters found by the algorithm.

On the other hand, the Clustering by Seeding algorithms ask the user to provide in input the labels of some data of the entire dataset. CobrasTS [11] is one of the most used Active Learning algorithms. This algorithm allows to create connections between the time series in the dataset asking the user if two time series that belong to two overlapping clusters are similar or not. The choice of the user leads to the creation of linking constraints between the two time series. Therefore, in this algorithm the user has the opportunity to personalize the cluster only via the tuning of the time series similarity.

Seeded KMeans[2] belongs to the Clustering by Seeding methods. The latter asks the user to provide as input the number of clusters and to indicate a subset of the data that should be clustered together. Therefore, in this case the user is not enabled to intervene in the middle of the computation, and only acts at the very beginning. Indeed, he provides information about how he wants to build the final clustering and the algorithm leverages this input for providing an output cluster that reflects the initial subset provided.

Among the unsupervised clustering algorithms, kShape[6] is certainly the most recent one. It computes the most representative time series in a given cluster and adds each new time series to one of these clusters based on the distance between time series. Being unsupervised, it requires no user input and interaction.

In this paper, we leverage prior work on feature-based time series clustering. In that prior work [9, 10], however, the tuning aspects of the clustering pipeline showing the user intervention by means of suitable parameters as well the significance of each step of the pipeline and its experimental assessment was not addressed, as we thoroughly do in this work.

3 HUMAN-CENTERED PARAMETERS

In this section, we discuss the human-in-the-loop aspects of FeatTS, by focusing on the parameters that can be changed along the pipeline as shown in Figure 1. With FeatTS the user has more opportunities to influence the clustering process by setting a number of parameters within the scope of a flexible pipeline. Moreover, as shown in the [9], the user can decide to tune the parameters at the end of each step and visualize the output to appreciate their effect on the system.

3.1 Learning Threshold

The personalization of the features based on the user needs is the first problem faced by FeatTS.

Indeed, with a high amount of supervision, the algorithm increases the relevance of the features that best correspond to the labels provided as input. Higher amounts of supervision lead to obtain as a result clusters that are more similar to the labels provided on input, thus with a higher quality in terms of obtaining results.

By opposite, whenever using a small or zero amount of supervising, the algorithm will favor the features based on their variance. In this case, the obtained clusters are less aligned with the input labels, but they can be better coupled with the raw data (i.e. the original time series).

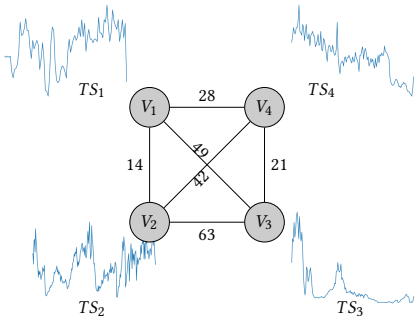
A parameter called 'Learning Threshold' will help the user to choose the right amount of supervision. The solution proposed for this problem represents the first two steps of Figure 1.

The first step is the extraction of the features from the time series that represent their peculiar characteristics, such as Mean, Median, Number of Peaks, etc. This process of extraction of the features can excerpt numerous characteristics of the time series but most of them lack the necessary quality. Therefore, the feature selection becomes pivotal in this stage. In particular, we compute the relevance of the extracted features using the feature values corresponding to the labeled portion of the time series.

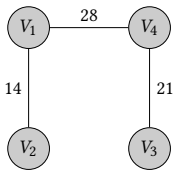
The Benjamini-Yekutieli is a supervised procedure that allows us to identify the relevance of the features, based on the labels associated with the time series. In this procedure, FeatTS provides as input the supervised subset of labels extracted via the Learning Threshold parameter.

The output of Benjamini-Yekutieli procedure is a list of features ranked by their p-values. The p-value is an important metric that allows us to quantify the significance of each feature.

Indirectly, this procedure also affects step 2 of the pipeline. Indeed, through the computation of the significance computed by the

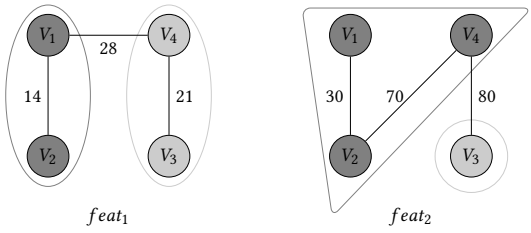


(a) Edge-weighted graph with distances as weights.



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

(b) The graph for a single feature after the cutting threshold



(c) Application of Community Detection algorithm for each feature.

Dataset	TS_1	TS_2	TS_3	TS_4
TS_1	1	$1 \cdot w_{12}$	0	$\frac{1}{2} \cdot w_{14}$
TS_2	$1 \cdot w_{21}$	1	0	$\frac{1}{2} \cdot w_{24}$
TS_3	0	0	1	$\frac{1}{2} \cdot w_{34}$
TS_4	$\frac{1}{2} \cdot w_{41}$	$\frac{1}{2} \cdot w_{42}$	$\frac{1}{2} \cdot w_{43}$	1

(d) Co-Occurrence Matrix

Figure 2: Illustration of the steps.

procedure, FeatTS adopts an algorithm of feature selection called PFA (Principal Features Analysis) that permits to compute a precise number of features with an acceptable relevance.

PFA is a variation of Principal Component Analysis (PCA). The key difference is that PFA preserves the original values of the features and thus the distance between them. Thus, we can leverage the concept of explaining variance, representing the ratio between the variance of one single feature and the sum of variances of all individual features, in order to obtain the exact number of features to be adopted in the process.

Figure 1 shows that the Learning Threshold parameter has an impact on step 1 and step 2 of the pipeline. In particular, the Benjamini-Yekutieli procedure is going to leverage the labels provided by the user during the supervised phase in order to provide the importance of each feature. Subsequently, the PFA leverages the importance of the features in order to select the subset of features.

3.2 Cutting Threshold

The primary goal of any cluster algorithm is to detect the similarity between the input data points. At this stage, we want to leave to the users the possibility of playing with the quality of the similarity by using a parameter that allows them to increase or decrease the similarity between the various time series.

In order to solve this problem, we chose to extract and evaluate the global relationships between the time series. This solution allows us to transpose the problem to another domain, i.e. to the graphical domain.

Therefore, for each feature chosen by PFA, 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 represent the distances computed by subtracting the absolute values of the feature of two connected time series. After computing all these distances, we create a fully connected graph network for each feature as shown in Figure 2a.

These graphs will be used to extract the global relationships between the time series. However, in order to obtain the level of similarity desired by the user, we will need to eliminate some arcs between the nodes.

This task will be done using the 'Cutting Threshold' parameter and the solution proposed for this problem represents the third step of Figure 1. The value of this parameter indicates the percentage of lower distances to be kept for each feature encoded as a graph.

Therefore, if the user wants to obtain small clusters formed by time series with a high similarity, they have to delete several arcs within the graphs. Hence, they have to choose a low value of the cutting threshold. On the other hand, if the aim is to obtain large clusters at the expense of similarity, in this case the user should keep as many arcs as possible. For this reason, a high cutting threshold is preferred. As an example, in Figure 2b we decide to keep the 50% of the edges for the f_1.

Once the edges are pruned, FeatTS extracts the actual global relationships through a process called Community Detection. It creates communities of vertices in the graphs based on the edges for each feature extracted from PFA, as shown in Figure 2c.

3.3 Number of Clusters

In classical semi-supervised clustering algorithms, the number of clusters is directly extracted using the classes provided as input by the user. This solution is limited since the user who might be interested in finding a different number of clusters from those in the supervised dataset cannot really obtain those through the clustering step. We show that the third parameter allows to obtain this behavior. Therefore, with the third parameter, it will be possible to choose the number of final clusters that the user wants to visualize.

The importance of this parameter is twofold. The first meaning is the possibility of finding undefined classes in the supervised dataset, providing additional information about the input dataset and further insights, as we will be showing in the experiments in Section 4. Moreover, this parameter combines all the communities of the global relations into a precise number of clusters. Indeed,

the number of communities detected may be different in each feature. Therefore, by defining a precise number of clusters, it favors the relationships obtained from a feature that has a number of communities equal to the number of clusters.

Thus, the next step of the pipeline needs to organize all the communities detected in a single data structure. 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 into 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 and each cell in the matrix corresponds to the similarity between time series in row and column. 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. Indeed, as shown in Figure 2d, we have computed the Co-Occurrence Matrix on the communities detected in Figure 2c. The similarity between two equal time series is always 1, while the similarity between the other time series is computed dividing the number of times that the two time series are in the same communities by the number of features. These values are multiplied by a weight that depends on the number of clusters that the user has chosen. Notice that the communities extracted from the graphs are not overlapping. For that reason, the Co-Occurrence Matrix is fully asymmetrical.

Thus, if the users want a high number of clusters, they will favor the features with a high number of communities. The resulting clusters will be composed of a restricted number of time series, but with a high similarity. Viceversa, a small number of clusters requested will favor the features with a small number of communities. In this case, the resulting clusters will be composed by a high number of time series but with a low similarity.

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 in order to apply the standard K-Medoid [7] algorithm to extract clusters of time series that have the smallest distance among them.

4 EXPERIMENTAL STUDY

In this section, we have shown the impact of the tuning of the three parameters on the clustering pipeline. We use real-life 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. Thus, it is very important to understand when a patient needs medical treatment before the GFR reaches its lowest value.

The usage of the real-time series supports the understanding of the importance of the humans in the entire machine learning process. Moreover, the possibility to interact with the algorithm, via the tuning of some parameters, could help the humans understand the underlying process and its results.

We have tested also the system robustness to the various parameters also for other benchmarking datasets adopting the UCR Time Series Archive[4]. All the experiments have been executed on a Server running Linux with 64GB of RAM, Intel Xeon CPU Skylake, IBRS @ 2.6GHz.

4.1 Probing the Learning Threshold

In this section, we will evaluate how FeatTS behaves when the Learning Threshold changes. In this experiment, we consider the quality of the resulting clusters by comparing the original labels of the dataset with those obtained from the final clusters. In order to have a measure for the quality of the clusters, we use a metric called Adjusted Mutual Information (AMI) [8].

Therefore, the purpose of this experiment is to evaluate how the quality of the resulting clusters increases as the learning threshold increases as well. This operation will be repeated for each value of the threshold. In our case, we decided to increase the threshold by 10% starting from 10% up to 90%.

In addition, in this experiment we set the Cutting Threshold to 80% and we set the number of clusters equal to those required within the supervised dataset.

Therefore, by modifying the amount of supervision and then consequently increasing the Learning Threshold, we show how the quality of the clustering process can be improved. Indeed, as shown in Figure 6, the quality of the resulting clusters increases where the Learning Threshold increases. However, this increase stops when we choose to use only 40% of the labels. This happens because from 50% onwards, the features chosen by PFA are equal, therefore the results obtained by FeatTS are the same. As shown also in the experiments in Figure 7 on other datasets, even at low threshold values, the same features can be selected as at high threshold values. This is often due to a small number of features that can actually be used. For space reasons, we omit the other results on UCR benchmarking datasets and make them available at an external link [1].

4.2 Probing the Cutting Threshold

In this section, we evaluate how the similarity of the time series within community detection varies based on the value of the Cutting Threshold provided by the user. Based on the explanation in Section 3.2, the similarity between time series should decrease as the percentage of the Cutting Threshold increase.

In order to prove this statement, we extract all the communities from the features chosen by PFA. Subsequently, we compute all the distances between the time series that belong to each community by leveraging DTW [3]. These distances will then be combined into an average for each community. This computation is shown in the Formula 1 where M_{c_k} represents the average of all the time series TS that belong to a single community C_k.

$$M_{c_k} = \forall\, TS_i, TS_j \;\in\; C_k,\; \frac{\sum_{i,j=0}^{|C_k|} DTW(TS_i, TS_j)}{\frac{|C_k| \cdot |C_k - 1|}{2}} \tag{1}$$

[1] https://cutt.ly/UmyE10Y

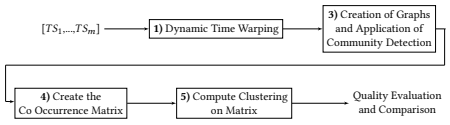


Figure 3: DTW Pipeline

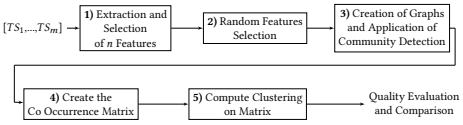


Figure 4: Random Features Pipeline

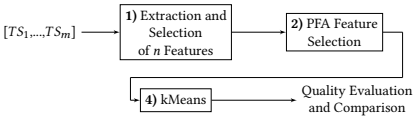


Figure 5: kMeans Ablation Pipeline

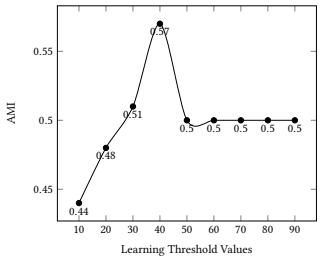
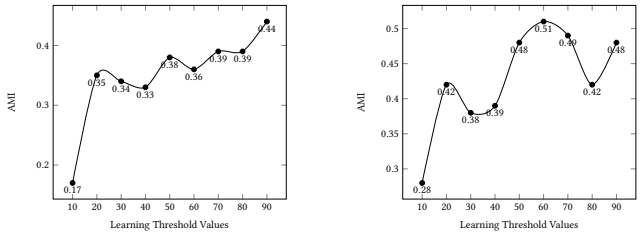
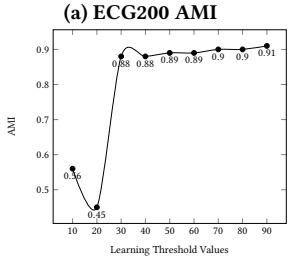


Figure 6: AMI for each Learning Threshold.

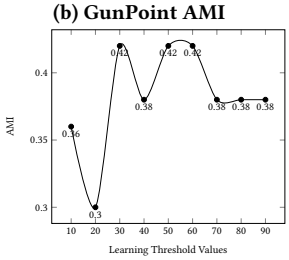


(a) ECG200 AMI

(b) GunPoint AMI



(c) Coffee AMI



(d) UMD AMI

Figure 7: AMI on UCR Datasets.

The obtained average of the single community will be combined again into the average of the other communities for the features chosen by PFA.

Therefore, assuming that C is the set of all the communities found for all the features, we compute the final average as in formula 2. This final average called M_{th} corresponds to the distance average of the chosen threshold.

$$M_{th} = \forall\, C_k \in C,\ \frac{\sum_{k=0}^{|C|} M_{c_k}}{|C|} \tag{2}$$

In this experiment, we set a percentage of Learning Threshold of 20% and we set the number of clusters equal to those required within the supervised dataset.

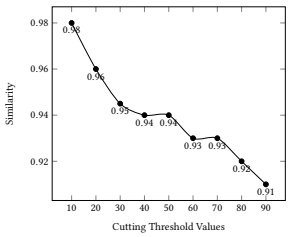
Table 8a shows that the average of the distances between the time series belonging to each community on the GFR Dataset increases as the Cutting Threshold increases. The increasing of the threshold corresponds to the decreasing of the similarity. Indeed, in Figure 8b, we can see that the drop of the similarity is about 7%. In order to compute the similarity, we have applied the formula 3. Indeed, assuming that TS_i and TS_j are two time series that belong to the dataset, we have normalized the average distances of each threshold M_{th} with the maximum distance found between two time series

$$S_{th} = 1 - \frac{M_{th}}{max(DTW(TS_i, TS_j))} \tag{3}$$

We report in Figure 9 some results for UCR datasets that are compatible with those obtained with the GFR dataset and show a similar trend. For space reasons, other results are omitted and can be found at the following link [2] .

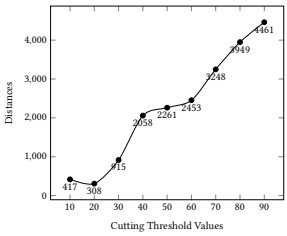
Threshold	Average Distances
10%	2449
20%	3060
30%	3966
40%	4366
50%	4354
60%	4625
70%	4648
80%	4661
90%	4662

(a) Average Distances

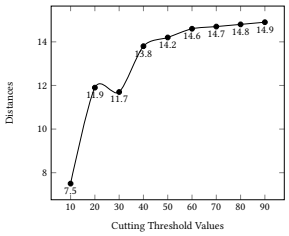


(b) Similarity

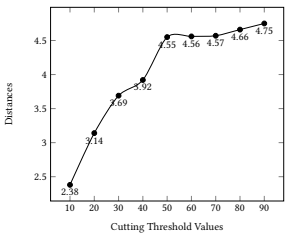
Figure 8: Similarity and distances within the Communities.



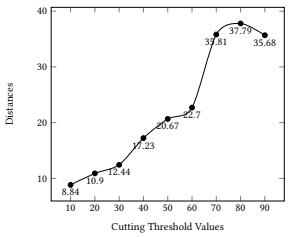
(a) Plaid Distances



(b) Coffee Distances



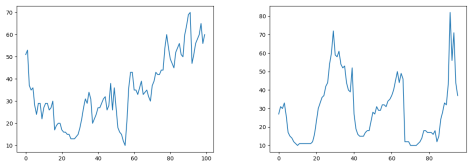
(c) Meat Distances



(d) UMD Distances

Figure 9: Distances within the Communities.

[2] https://cutt.ly/YmyE6PL



(a) Acute Dialysis (b) Chronic Dialysis

Figure 10: Time series of the GFR signal for patients treated for Acute and Chronic Dialysis.

4.3 Probing the Numbers of Clusters

In this experiment, we show the importance of deciding upon the number of clusters to be extracted despite a different number of labels provided as input. The GFR dataset shows two classes called Kidney Failure and Not Kidney Failure. With the help of a domain expert, we have increased the number of classes to analyse the behaviour of FeatTS with classes different from those given as input. After interacting with the expert, we obtained out of the Kidney Failure class two other classes, namely Chronic Dialysis and Acute Dialysis.

Patients under chronic dialysis need to go regularly at the hospital for health care. Chronic kidney disease is a kidney disease in which there is gradual loss of kidney function over a period of months to years. Acute kidney injury is an abrupt loss of kidney function that develops within 7 days. Basically, an acute kidney injury occurs when the kidneys are exposed to something harmful, or which is considered harmful by the body.

In this experiment, we show the usefulness of FeatTS to help the medical doctors separate the cases of chronic dialysis from the acute dialysis while not having these labels available as input. In Figure 10 we show the two classes discovered by FeatTS.

5 ABLATION TEST

In this section, we present a handful of ablation tests devoted to showing the importance of each step of the FeatTS pipeline. These tests were performed on several datasets belonging to the UCR.

DTW versus Features. The idea behind this ablation test is to show the importance of adopting the distance between the features instead of using the distance between the raw time series using Dynamic Time Warping (DTW) [3]. In a nutshell, this corresponds to removing step 1 and 2 from the pipeline in Figure 1, i.e. the extraction and the selection of the features using PFA, as shown in Table 1. The latter shows the results obtained by replacing the distances between the features with the distance computed by the DTW directly on the time series. The results are expressed in the column called DTW.

The obtained results showed that, 14 datasets (out of 15) have a worse behavior if DTW is adopted. Indeed, only in one dataset (*OliveOil*) the usage of DTW outperforms the usage of the features. On average, we have a difference in terms of AMI of 0.31. This ablation test confirms the importance of using distances between features instead of merely using distances between raw data.

P-value versus Random. Table 1 shows the results removing the ordering of the features based on their relevance, as shown in Figure 4. We repeated this test 5 times and we averaged the results. The column Rand represents the average of the results obtained by FeatTS using Random Features.

The results in Table 1 show that the performance of the algorithm drastically deteriorates for the majority of the datasets (13 out of a total of 15 datasets) if random features are employed. Indeed, computing the average overall the results obtained by the two experiments, we have a difference of 0.35 in terms of AMI. Therefore, the ordering of the features based on their relevance turns out to be indispensable in order to achieve good results.

kMeans versus FeatTS. The purpose of this ablation test is to show the importance of capturing the global relationship among the raw time series samples through graph encoding and of the subsequent application of the Community Detection algorithm as in this approach. Therefore, we replace step 3,4 and 5 as in Figure 1 with k-Means, i.e. a classical clustering algorithm in its multidimensional version. Hence, once the features have been extracted and selected through PFA in step 1 and 2, we apply the k-Means algorithm to obtain the clustering, as shown in Figure 5.

In Table 1, we show the results obtained capturing the global relationship among the raw time series samples through graph encoding and of the subsequent application of the Community Detection. The column k-Means shows the results obtained by k-Means among the features selected by each dataset used in this experiment. We have highlighted in bold the best results obtained between k-Means and FeatTS for each dataset.

The results show that FeatTS outperforms k-Means in the majority of the cases. Indeed, there are only four datasets for which k-Means shows slightly better results, namely UMD, $Meat$, $Coffee$ and *OliveOil*. On average, the results obtained by FeatTS outperforms the results obtained by k-Means of 0,08 in terms of AMI.

This ablation test, therefore, shows the importance of capturing the global relationships between the various time series in order to achieve better results in terms of performance.

Dataset	FeatTS	DTW	Rand	k-Means
ScreenType	**0,02**	0,01	0	0,01
UMD	0,3	0,27	0,01	**0,42**
TwoLeadECG	**0,88**	0	0,02	0,75
ECG200	**0,32**	0,08	0,06	0,16
Computers	**0,09**	0	0	0
Coffee	0,8	0,01	0,12	**0,9**
GunPoint	**0,56**	0	0	0,26
Arrowhead	**0,28**	0,2	0,02	0,24
ItalyPowerDemand	**0,57**	0	0	0,51
Meat	0,42	0	0,16	**0,48**
OliveOil	0,15	0,32	0,2	**0,35**
Plaid	**0,35**	0,01	0,11	0,03
Asphalt Regularity	**0,54**	0	0	0,35
Asphalt Obstacles	**0,38**	0,27	0,02	0,37
Gesture Pebble	**0,25**	0,02	0	0,16

Table 1: Ablation Test Results.

6 CONCLUSION AND FUTURE WORK

In this article we have used FeatTS to showcase the human-centered aspects of a clustering pipeline. To this end, three parameters (namely 'Learning Threshold', 'Cutting Threshold' and 'Number of clusters') have been introduced in the clustering pipeline and probed through an extensive analysis.

This work is the first to make the user aware of the choices made in a clustering step and to make the latter process transparent. Future work can be devoted to introduce more tuning, by studying the impact of the selected features on the final results and make the latter more explainable.

REFERENCES

[1] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. 2015. Time-series clustering–a decade review. *Information Systems* 53 (2015), 16–38.

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

[3] Donald J. Berndt and James Clifford. 1994. Using Dynamic Time Warping to Find Patterns in Time Series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining* (Seattle, WA) *(AAAIWS'94)*. AAAI Press, 359–370.

[4] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. 2018. The UCR Time Series Classification Archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[5] T Warren Liao. 2005. Clustering of time series data—a survey. *Pattern recognition* 38, 11 (2005), 1857–1874.

[6] John Paparrizos and Luis Gravano. 2016. K-Shape: Efficient and Accurate Clustering of Time Series. *SIGMOD Record.* (2016).

[7] Hae-Sang Park and Chi-Hyuck Jun. 2009. A simple and fast algorithm for K-medoids clustering. *Expert systems with applications* 36, 2 (2009), 3336–3341.

[8] Simone Romano, Nguyen Xuan Vinh, James Bailey, and Karin Verspoor. 2016. Adjusting for chance clustering comparison measures. *The Journal of Machine Learning Research* 17, 1 (2016), 4635–4666.

[9] 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)*. Association for Computing Machinery, 5 pages.

[10] Donato Tiano, Angela Bonifati, and Raymond Ng. 2021. Feature-driven Time Series Clustering. In *Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021*. OpenProceedings.org, 349–354.

[11] Toon Van Craenendonck, Wannes Meert, Sebastijan Dumančić, and Hendrik Blockeel. 2018. Cobras ts: A new approach to semi-supervised clustering of time series. In *International Conference on Discovery Science*. Springer, 179–193.

[12] Haishuai Wang, Qin Zhang, Jia Wu, Shirui Pan, and Yixin Chen. 2019. Time series feature learning with labeled and unlabeled data. *Pattern Recognition* (2019).

[13] Sicheng Xiong, Javad Azimi, and Xiaoli Z Fern. 2013. Active learning of constraints for semi-supervised clustering. *IEEE Transactions on Knowledge and Data Engineering* 26, 1 (2013), 43–54.