ML for time-series - MVA 2023/2024

Benjamin Lapostolle benjamin.lapostolle@polytechnique.edu
Jean-Baptiste Himbert jean-baptiste.himbert@polytechnique.edu

February 8, 2024

Abstract

This project aims to explore the research paper titled " Time-series Clustering via NMF in Networks"[4] by Guowang Du et al, which builds upon the foundational work presented in another paper[6]. It introduces a novel approach to time-series classification by transforming time-series data into a graph framework. In this graph, nodes are interconnected based on the distance metrics between time series. Guowang Du et al have proposed new methods for community detection leveraging Non-negative Matrix Factorization (NMF). Their methodology demonstrates enhanced classification results for most of the time series under consideration. Building upon their work, we replicate their finding, compare NMF methods to other variants such as Node2Vec and DeepNMF, and point out different practical shortcomings of the proposed framework.

1 Introduction and contributions

Time series data has been extensively studied due to its prevalence in numerous domains. Among the various processing tasks that can be performed, a common objective is to find clusters in a dataset. This has several applications, such as in medicine, where it can, for instance, aid in classifying normal brain behavior in contrast to that of a diseased patient.

Among the methods for time-series clustering, some involve extracting features from each time-series so that the existing algorithms for clustering static data can be directly used[15]. The approach chosen here only focuses on the distance between the time series. We derive distances between time series and employ them to construct a graph, with each node representing a distinct time series and connected to others based on their similarity. The intuition is that time series lying close to each other in the distance space will bunch up together, creating effective clusters in the graph. Analyzing neighboring relationships will thus allow us to extract global information on the time series.

Constructing the graph is a critical determinant of the clustering process outcomes. Following the original paper, we use the Distance Time Warping (DTW) distance[8] to build it, which is a well-known measure for comparing time series. Subsequently, we connect nodes for which the distances are below a certain threshold ϵ. The selection of this threshold holds significant importance because its value can either isolate nodes extensively or connect them all. As we will explore further, its value greatly influences the outcomes of this method. Then, we use an algorithm of community detection to cluster the time series. The paper we studied used Negative Matrix Factorization (NMF) to perform the time-series clustering. NMF is a powerful tool for data analysis, especially in high dimensional settings, enhancing interpretability by approximating the target matrix as the product of two low-rank matrices. It's particularly effective in real-world problems with non-negative constraints, offering fast convergence and the authors achieved better accuracy in comparison with the methods employed in [6].

In addition to classic clustering algorithms (*KMeans*, *GaussianMixture*, and *Spectral Clustering*), we compare NMF methods with two deep variants called DeepNMF[7] and Deep Autoencoder-like NMF (DANMF)[5]. Originally developed for hierarchical clustering applications, they approximate the feature matrix with sequential NMF. Finally, we also include two prominent approaches in network embedding called

Node2Vec[9] and Large-scale Information Network Embedding (LINE) [17]. Node2Vec leverages architectures that have demonstrated effectiveness in Neural Language Processing tasks by generating a vocabulary of random walks originating from the graph. LINE is an algorithm more suited to big graphs that implements negative sampling to decrease the computational cost and is designed to preserve both first-order and second-order proximity in a network.

After having tested all the methods on various data sources, we highlight some weaknesses of the framework regarding graph construction and evaluate its robustness to noise and outliers. Although the majority of the work was done together, Benjamin focused on the LINE and Node2Vec algorithms while Jean-Baptiste was tasked with all the NMF variants (shallow and deep). For the code, we used this repository for the LINE implementation and used scikit-learn for *KMeans*, *GaussianMixture*, *SpectralClustering*, and *vanilla NMF*. Otherwise, we followed the pseudocode available in the corresponding article and re-implemented all other methods using Python. Our code can be found in this repository.

2 Method

2.1 Network Construction

As stated before, we construct a network $G = (U, V)$ with U nodes and V edges. The nodes of the network are the time series for a given dataset and the nodes are connected if their similarity is lower than a threshold ϵ. In other words, for each $u_1, u_2 \in U$, $(u_1, u_2) \in V$ if and only if $d_{DTW}(u_1, u_2) < \epsilon$. The weight of the edges is equal to the normalized similarity between the two nodes to have values between 0 and 1. Any other distance measures can be used to build such a graph, however, we only focus on the DTW distance as it yields the best results [6].

2.2 Clustering Algorithm

2.2.1 Non-negative Matrix Factorization

The authors leverage different techniques of NMF to perform clustering. The overall idea of NMF is to decompose the matrix of data $X_{n\times p}$ into a product of two non-negative matrices $W_{n\times c}$ and $H_{c\times m}$, *ie* $X = WH$. The dimension c is much lower than the dimensions of the data matrix. In this decomposition, the column of the largest coefficient of the line i of the matrix W is the cluster the node belongs to.

Finding W and H amount to solve the optimization problem $\min_{W,H\geq 0}[L(X, WH) + P(W, H)]$, which is a non-convex problem. To solve it, common practice [14, 13, 10] optimizes W and H separately in an iterative manner, as the partial problems are all convex. L is a loss function that measures the quality of approximation, and "P" is an optional penalty function. L is typically either a least squares criterion (Frobenius norm of matrices) or the Kullback-Leibler (KL) divergence. P is an optional regularization penalty used to enforce desired properties of matrices W and H, such as the sparsity of the matrices or the smoothness of solutions.

2.2.2 Variants of NMF

To solve this optimization problem, many variants have been proposed. In this project, we also implemented:

1. *Sparse-NMF[11]*: In this variant, we introduce a penalty to have a more sparse solution and solve: $\min_{W,H\geq 0} \frac{1}{2}\|X - WH\|_F + \lambda \sum_{i=1}^{c} |H_{i,.}|$
2. *Semi-NMF[3]*: We remove the constraint $W \geq 0$ in the optimization problem, allowing this method to process matrices with negative coefficients.
3. *KNMF[18]*: KNMF is a more general version of NMF where we use the kernel trick to deal with non-linearities in the data. It consists of performing NMF on the kernel-transformed feature matrix, and we used in our experiments the RBF kernel.

4. *Sym-NMF[12]*: We impose W and H to be their symmetric counterparts and consider the following optimization problem: $\min_{W,H\geq 0} \frac{1}{2}\|X - HH^T\|_F$

All NMF methods employ a similar optimization process, described in the following pseudocode 1:

Algorithm 1 NMF-Based Clustering

Input: Adjacency matrix X, Cluster number K, Error Threshold τ, Maximum iteration *max_iter*
Output: cluster of each time-series

```
Initialize: Randomly initialize W and H
for iter = 1 to max_iter do
    Compute X = WH
    e = ||X − WH||_F
    if e < τ then
        Break
    Fixed W, update H
    Fixed H, update V
return the argmax of W for each line
```

2.3 DeepNMF and Deep Autoencoder-like NMF (DANMF)

Motivated by hierarchical examples featuring interleaving communities, researchers started to develop "deep" NMF variants[7, 5]. The idea is to approximate the feature matrix A via a series of non-negative matrix factorizations as such:

$$A \sim U_1 U_2 \dots U_p V_p \tag{1}$$

with $V_p \in \mathbb{R}_+^{k\times n}, U_i \in \mathbb{R}_+^{r_{i-1}\times r_i} with n = r_0 \geq r_1 \geq \dots \geq r_p = k$.

The full model is trained by first computing the NMF at every layer as such $A \sim U_1 V_1; V_1 \sim U_2 V_2; \dots; V_{p-1} \sim U_p V_p$, and then finetuning globally through the decoding loss function $\mathcal{L}_D = ||A - U_1 U_2 \dots U_p V_p||_F^2$ with positivity constraints.

The Deep Autoencoder-like model tries to improve on the method by also optimizing for two additional losses called the encoding loss $\mathcal{L}_E = ||V_p - U_p^T \dots U_2^T U_1^T A||_F^2$ with $V_p \geq 0, U_i \geq 0, \forall i = 1 \dots p$ and the regularization loss $\mathcal{L}_{reg} = \lambda tr(V_p L V_p^T)$ where L is the laplacian matrix and λ a parameter. This triple loss, inspired by Autoencoder models, aims to achieve better representation power. Once again, optimization is performed in two steps, first by computing sequential NMFs and then by a joint optimization procedure.

2.4 Node2Vec and LINE

As presented in the introduction, we used two more algorithms specific to graph data that create embeddings of the nodes: the Node2Vec and the LINE algorithm. LINE is particularly suited for large graphs. This implementation takes into consideration the intuition that two nodes connected and that share similar neighbors should be close in the embedding space by concatenating the embeddings of two LINE models that each preserve first and second-order proximity. Moreover, as computing the loss normally requires summing over all the sets of vertices, LINE introduces negative sampling to reduce the number of vertices involved in the computation.

Node2Vec creates embeddings by using models such as Skipgram models — heavily used for NLP tasks [16] — on a vocabulary of random walks on the graph. The random walks are parametrized by two values **p** and **q**, allowing them to control their drift from their starting node. **p** controls the probability of visiting the previous node, and **q** controls the probability of visiting the neighbors of the previous node. The authors in [9] demonstrated that setting higher return probability values in the random walk process leads to the formation of clusters consisting of neighboring nodes, which is a behavior that should align with our data.

3 Data

To evaluate the different clustering methods presented in Section 2, we used two types of data. The first part comes from the **UCR time-series Classification Archive** [2], a well-known repository hosting a variety of time-series datasets used for benchmark purposes. Keen on the idea of comparing our results with the ones from the original paper [4], we used datasets *Beef*, *CinCECGTorso*, *Coffee*, *DiatomSizeReduction*, *ECGFiveDays*, *MoteStrain*, *OliveOil*, *SonyAIBORobotSurface1*, *SonyAIBORobotSurface2*, *Symbols*, *TwoLeadECG*. As those datasets are fairly small and clean, they are particularly suited for evaluation purposes as they allow to test methods with very different types of time-series (medical signals, financial time-series, industrial signals ...)[1].

However, their small sizes made it difficult to assess how our methods were performing in larger-scale settings. To account for this issue, we leveraged the **PTB Diagnostic ECG Dataset** [1], which features 14,500 ECG signals from sane and sick individuals. These signals, sampled at 125 Hz, are padded to a length of 188. Some example signals can be seen in the Appendix in Figure 1.

Most of our data was clean, and already normalized, with little to no outliers and no missing data. As such, hard preprocessing steps were not necessary. However, to assess the robustness of clustering with NMF, we incrementally added noise and outliers to the PTB dataset in our third experiment. Starting from mostly clean data allowed us to have better control over the inconsistencies we added.

4 Results

Our experiments are threefold. We first assess each method on every dataset by comparing their Rand Score (as done in [4, 6]). Then, using the UCR datasets, we show the particular sensibility of the method to the choice of ϵ used to build the graph from the distance matrix. Finally, we assess the robustness of NMF clustering to noise and outliers on the PTB dataset. We followed the method described in section 2, and we performed a grid search to find the best epsilon value.

4.1 Main results

Table 1 gathers the main results of our experiments[2], which are the mean over 4 independent runs. Viewing from the results, it is difficult to distinguish a method better than all others. However, we can still note that Sym-NMF and DANMF are significantly behind almost all other methods. We think that Sym-NMF is limited by its few parameters compared to other NMF methods (W^T and H being equal) and that the DANMF additional optimization objective diverts the method away from the optimality when processing small datasets. However, for the PTB dataset (which is the largest), the bigger methods like DeepNMF and DANMF start to take over: their hierarchical structure allows them to find better optimum than classic NMF. A mention must be made for Sparse-NMF which, without any particularly complex initialization scheme, has the highest number of high scores of all the methods, and scales well with the size of the PTB dataset. This shouldn't be surprising, as it is known that methods inducing sparsity are very performant and efficient when size increases.

4.2 Sensibility to epsilon

While the NMF methods seem promising, we observed experimentally that the results varied strongly depending on the ϵ value used for building. During our grid search, we kept track of the best ϵ value and compared it to the distribution of distances in the dataset using quantiles to see if a pattern emerged. Figure 3 gathers all optimal ϵ and their respective quantile correspondence for each dataset-method combination[3]. As we can see from the plot (and the table), there are strong discrepancies in ϵ values across the different datasets, but, for the same dataset, also across the different clustering methods. This continues to be the case

[1] Additional information about each dataset can be found in Table 2 in Appendix.
[2] Additional visualization can be found in Appendix in Figure 2.
[3] Proper values can be found in Table 3 in the Appendix.

	Sony1	Sony2	Beef	CinC	Coffee	Diatom	ECG5	Mote	Olive	Symbols	2Lead	PTB
Node2Vec	**1.00**	0.74	0.78	0.65	0.86	0.82	0.60	0.66	0.83	0.96	0.56	0.59
LINE	0.73	**0.79**	0.73	0.66	0.8	0.92	0.60	0.75	0.87	0.87	0.62	0.52
Kmeans	0.81	0.74	**0.80**	0.66	0.80	0.88	0.64	0.66	0.87	0.96	0.60	0.52
GaussianM	0.81	0.74	0.79	**0.67**	0.80	0.93	0.62	0.81	0.87	**0.98**	0.60	0.52
Spectral	0.90	0.78	**0.80**	0.66	0.80	0.93	0.64	0.73	0.86	0.96	0.60	0.52
NMF	0.73	0.78	0.76	0.66	0.75	0.95	0.64	0.81	**0.88**	**0.98**	**0.70**	0.58
Semi-NMF	0.90	0.77	0.75	0.66	0.80	**1.00**	0.63	0.66	0.81	0.95	0.60	0.53
Sym-NMF	0.66	0.64	0.74	0.66	0.61	0.75	0.60	0.66	0.65	0.76	0.60	0.51
KNMF	0.90	0.76	0.74	0.65	0.80	0.82	**0.64**	0.81	0.86	0.91	0.60	0.52
Sparse-NMF	0.81	**0.79**	0.71	0.66	**0.93**	0.82	**0.65**	**0.90**	0.86	0.96	0.60	0.61
DeepNMF	**1.00**	**0.79**	0.57	0.61	0.51	0.56	0.56	0.49	0.69	0.62	0.60	0.60
DANMF	0.81	0.54	0.57	0.59	0.49	0.56	0.60	0.49	0.69	0.62	0.53	**0.62**

Table 1: Rand Score for different dataset-method combination

when translating in the correspondent quantile (taken from the distance values of the similarity matrix). For example, the Sony1 dataset has a very high optimal epsilon value for *KMeans*, *GaussianMixture*, *NMF*, *Semi-NMF*, and *Sparse-NMF*, while having a really low value for the other methods. This is a major issue, as ϵ calibration is therefore needed before applying the framework to data, which is unpractical with real data.

4.3 Robustness to noise and outliers

As a final experiment, we wanted to assess the robustness of the NMF clustering method to noise and outliers. With this goal in mind, we added white noise of standard deviation σ to all time-series of the PTB dataset and outliers taken at random[4] to replicate real time series. We then performed the usual framework and applied the NMF algorithm to cluster the data. Figure 4 and Table 5-4 gather the rand score of experiments with different noise and outliers magnitude.

As it was expected, performance drops swiftly when increasing noise and outlier magnitude. Indeed, the time series correspond to ECG data, for which both big and small variations are important for diagnosis. However, it may be noted that the method is a little bit more resilient to outliers than to noise. This comes from the fact that the DTW distance is not robust to noise, while its flexible matching can mitigate outlier influence.

Conclusion

In this paper, we build upon the methodology developed by Ferreira et al and Du et al to cluster time series using community detection tools in networks. We implemented a vast array of NMF methods (including deep methods) and compared them to other classic clustering algorithms and tools from graph clustering (Node2Vec and LINE) on both small and large-scale datasets and found that, while all methods performed evenly on small datasets, deep and sparsity-inducing methods scaled better with size. Finally, we pointed out several limitations of our framework, mainly the reliance on the epsilon chosen for the graph and the presence of noise and outliers.

References

[1] Kreiseler D. Schnabel A. Bousseljot, R. Nutzung der ekg-signaldatenbank cardiodat der ptb über das internet. *Biomedizinische Technik / Biomedical Engineering*, pages 317–318, 1995.

[4] To add outliers, we added to every point, with probability $p = 0.01$, a value taken uniformly between 0 and *magnitude*.

[2] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, July 2015. www.cs.ucr.edu/~eamonn/time_series_data/.

[3] Chris H.Q. Ding, Tao Li, and Michael I. Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):45–55, 2010.

[4] Guowang Du, Lihua Zhou, Yuan Fang, and Ming Yang. Time series clustering via nmf in networks. In *2018 IEEE 16th International Conference on Software Engineering Research, Management and Applications (SERA)*, pages 87–92, 2018.

[5] Zibin Zheng Fanghua Ye, Chuan Chen. Deep autoencoder-like nonnegative matrix factorization for community detection.

[6] Leonardo N. Ferreira and Liang Zhao. Time series clustering via community detection in networks. *Information Sciences*, 326:227–242, January 2016.

[7] Jennifer Flenner and Blake Hunter. A deep non-negative matrix factorization neural network. *Semantic Scholar*, 2017.

[8] Toni Giorgino. Computing and visualizing dynamic time warping alignments in r: The dtw package. *Journal of Statistical Software*, 31(7):1–24, 2009.

[9] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, 2016.

[10] Lee Hyekyoung, Jiho Yoo, and Seungjin Choi. Semi-supervised nonnegative matrix factorization. *Signal Processing Letters, IEEE*, 17:4 – 7, 02 2010.

[11] Jingu Kim and Haesun Park. Sparse nonnegative matrix factorization for clustering. Technical report, Georgia Institute of Technology, 2008.

[12] Da Kuang, Chris Ding, and Haesun Park. *Symmetric Nonnegative Matrix Factorization for Graph Clustering*, pages 106–117.

[13] Daniel Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.

[14] Seung H. Sebastian Lee, Daniel D. Learning the parts of objects by non-negative matrix factorization. *Nature*, 1999.

[15] T. Warren Liao. Clustering of time series data - a survey. *Pattern Recognition*, 38(11):1857–1874, 2005.

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26:3111–3119, 2013.

[17] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web (WWW '15)*. ACM, 2015.

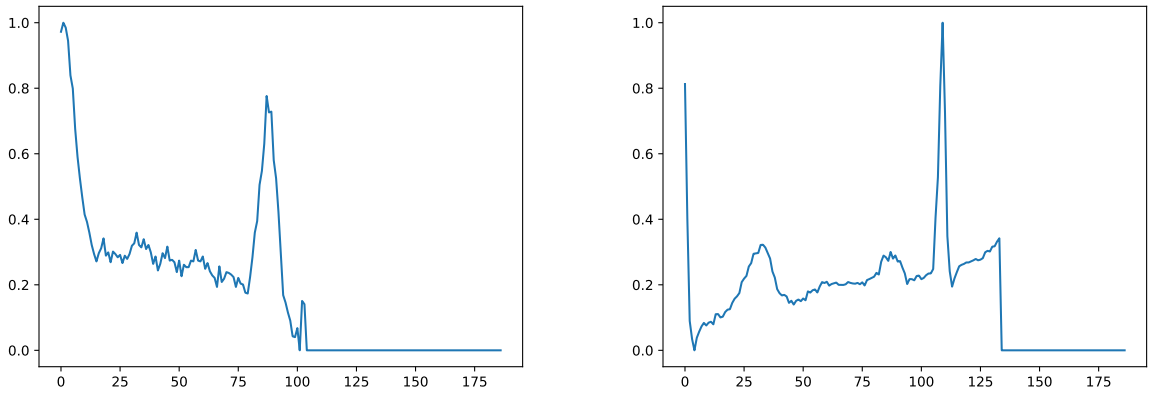
[18] Daoqiang Zhang, Zhi-Hua Zhou, and Songcan Chen. Non-negative matrix factorization on kernels. volume 4099, pages 404–412, 08 2006.

Appendix

4.4 Datasets

Name	Number Classes	Size	time-series Length
Beef	5	30	470
CinCECGTorso	4	40	1639
Coffee	2	28	286
DiatomSizeReduction	4	16	345
ECGFiveDays	2	23	136
MoteStrain	2	20	84
OliveOil	4	30	570
SonyAIBORobotSurface1	2	20	70
SonyAIBORobotSurface2	2	27	65
Symbols	6	25	398
TwoLeadECG	2	23	82

Table 2: Description of used UCR datasets



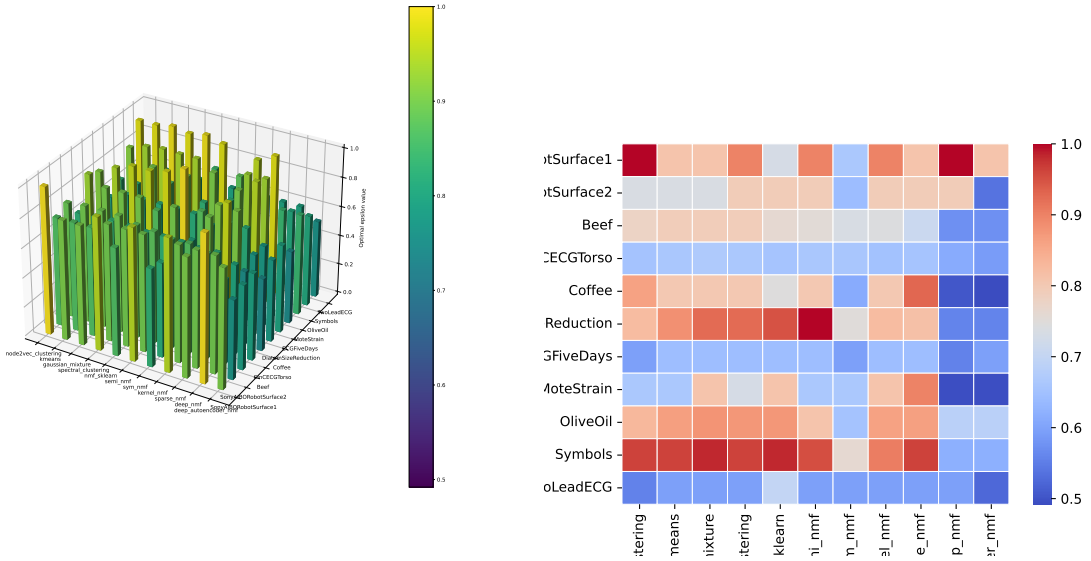
(a) Example 1

(b) Example 2

Figure 1: Example of time-series from the PTB dataset

4.5 Experiments

4.5.1 Performance evaluation



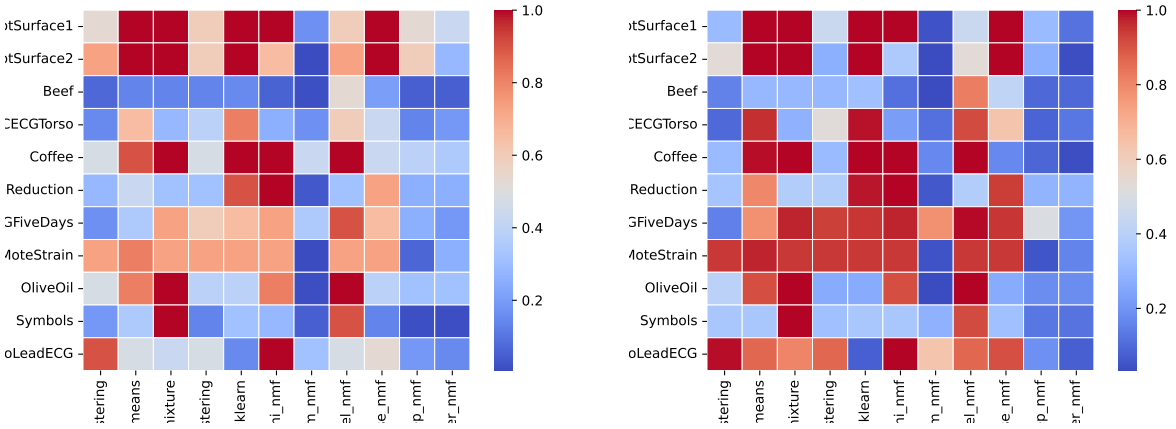
(a) 3D bar graph (b) Heatmap

Figure 2: Rand Score plots for all dataset - algorithm combination

4.5.2 Epsilon

	Sony1	Sony2	Beef	CinC	Coffee	Diatom	ECG5	Mote	Olive	Symbols	2Lead
Node2Vec	0.53	0.73	0.07	0.15	0.48	0.28	0.17	0.73	0.48	0.19	0.90
LINE	0.7	0.8	0.4	0.3	0.5	0.3	0.3	0.4	0.4	0.3	0.5
Kmeans	1.00	1.00	0.14	0.66	0.90	0.43	0.35	0.81	0.81	0.35	0.48
GaussianM	1.00	1.00	0.14	0.28	1.00	0.31	0.73	0.73	1.00	1.00	0.43
Spectral	0.59	0.59	0.14	0.39	0.48	0.31	0.59	0.73	0.39	0.14	0.48
NMF	1.00	1.00	0.15	0.81	1.00	0.90	0.66	0.73	0.39	0.31	0.15
Semi-NMF	1.00	0.66	0.06	0.25	1.00	1.00	0.73	0.73	0.81	0.28	1.00
Sym-NMF	0.17	0.01	0.02	0.17	0.43	0.03	0.35	0.01	0.01	0.05	0.31
KNMF	0.59	0.73	0.53	0.59	1.00	0.31	0.90	0.73	1.00	0.90	0.48
Sparse-NMF	1.00	1.00	0.21	0.43	0.43	0.73	0.66	0.73	0.39	0.14	0.53
DeepNMF	0.53	0.59	0.05	0.14	0.39	0.25	0.25	0.06	0.31	0.01	0.19
DANMF	0.43	0.28	0.05	0.19	0.35	0.25	0.19	0.25	0.31	0.01	0.15

Table 3: Optimal epsilons for different dataset-method combination



(a) Optimal Epsilon

(b) Correponding quantile

Figure 3: Optimal Epsilon and corresponding quantile for all dataset - algorithm combination

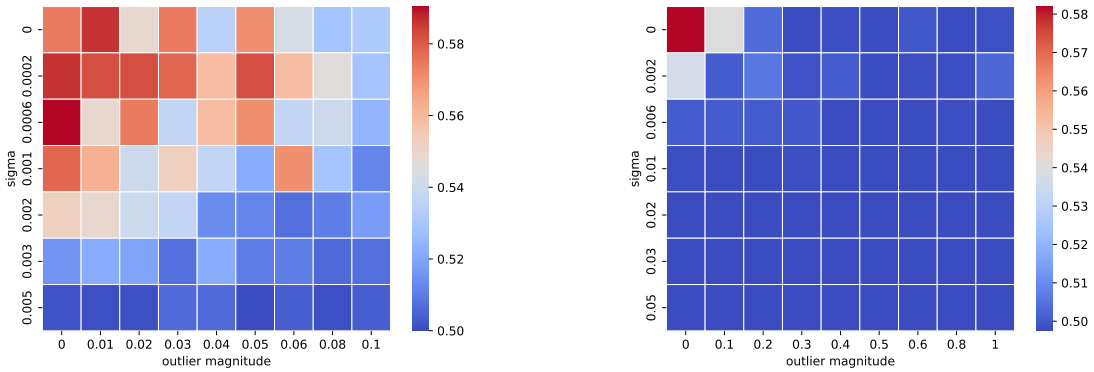
4.5.3 Noise and Outliers

Outlier Magnitude	0	0.01	0.02	0.03	0.04	0.05	0.06	0.08	0.1
$\sigma = 0$	0.57	0.59	0.55	0.57	0.53	0.57	0.54	0.53	0.53
$\sigma = 2e-4$	0.59	0.58	0.58	0.58	0.56	0.58	0.56	0.55	0.53
$\sigma = 6e-4$	0.59	0.55	0.57	0.54	0.56	0.57	0.54	0.54	0.52
$\sigma = 1e-3$	0.58	0.56	0.54	0.55	0.54	0.52	0.57	0.53	0.51
$\sigma = 2e-3$	0.55	0.55	0.54	0.54	0.51	0.51	0.51	0.51	0.52
$\sigma = 3e-3$	0.52	0.52	0.52	0.51	0.52	0.51	0.51	0.51	0.51
$\sigma = 5e-3$	0.50	0.50	0.50	0.51	0.51	0.50	0.50	0.50	0.50

Table 4: NMF Rand Score on the PTB dataset for different magnitudes of noise and outliers (small parameters)

Outlier Magnitude	0	0.1	0.2	0.3	0.4	0.5	0.6	0.8	1.0
$\sigma = 0$	0.58	0.54	0.50	0.50	0.50	0.50	0.50	0.50	0.50
$\sigma = 2e-3$	0.54	0.50	0.51	0.50	0.50	0.50	0.50	0.50	0.50
$\sigma = 6e-3$	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
$\sigma = 1e-2$	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
$\sigma = 2e-2$	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
$\sigma = 3e-2$	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
$\sigma = 5e-2$	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50

Table 5: NMF Rand Score on the PTB dataset for different magnitudes of noise and outliers (big parameters)



(a) Small parameters

(b) Big parameters

Figure 4: Rand score of NMF method for different magnitudes of added noise/outliers using small and big parameter values