Clustering of time series using topological data analysis

Pilyugina Polina, Shvetsov Nikolay, Vlasov Andrei, Talitsky Alexandr

21 March 2019

1 Introduction

Topological data analysis (TDA) is an approach to analyzing datasets, which allows to extract and analyze their underlying shape and structure. This approach is growing popularity among researches, as it allows to work with high-dimensional and complex datasets, and it is robust to small perturbations in data. therefore researcher try to include TDA in their pipelines for solving different data analysis problems. Several papers were published on the application of TDA to time series analysis, and in this project we mostly focus on papers written by Seversky et al. (2016), Gidea et al. (2018) and Umeda (2017). However, their approaches mainly focus on the analysis of time series itself, while we, on the contrary, seek to use TDA for clustering of commercial time series. Therefore this project is focused on the application of approach, presented by Lacombe et al. (2018), to the clustering of time series with topological data analysis. In this project we seek to recreate the algorithm of Lacombe, more closely examine how it works and include it in our pipeline for clustering of commercial time series.

The problem of clustering of time series is itself very tricky, and TDA, to our knowledge, has never previously been applied to the problem of time series clustering, specifically, to the clustering of commercial time series. Therefore no coding solutions exist by now, and thus all algorithms need to be implemented from scratch. In addition, very little papers were written on the related topics. For this project we create persistence diagrams of time series, following the approach of Seversky et al. (2016), and then apply Lacombe's algorithm to approximate optimal transport and barycenters in the space of persistence diagrams. This approximated distances will afterwards be used in k-means and agglomerative clustering.

One drawback of TDA is that it is rather computationally intense and requires a lot of hardware resources and time, that is why the approach of Lacombe was used, because it allows for parallelization, as it uses iterative methods, and also it uses convolutions for matrix multiplication, which allows to reduce memory usage.

Therefore the pipeline of the approach, used in this project, can be summarized by the following chart:

Point cloud creation	→	**Rips complex**	→	**Persistence diagrams**	→	**Optimal transport**	→	**Clustering of PDs**

Table 1: Pipeline

2 Literature review

In case of TDA, the most popular method of analysis is creation of persistence diagrams and their further analysis. In the paper by Seversky et al. (2016) the approach on how to apply TDA to time series is described. This approach is shared by most papers on application of TDA to time series. The idea is to create a point cloud from the time series and then apply Rips filtrations to create persistence diagram. From the theoretical point of view, Rips simplical complexes of point clouds — $R(Z, \epsilon)$, were obtained as follows:

- for each $k = 0, 1, 2, ...,$ a k-simplex of vertices $\{x_{i1}, ..., x_{ik}\}$ is a part of $R(X, \epsilon)$ iff the mutual distance between any pair of its vertices is less than ϵ, that is:

$$d(x_{ij}, x_{il}) < \varepsilon, \text{ for all } x_{ij}, x_{il} \in \{x_{i1}, ..., x_{ik}\} \tag{1}$$

Then this Rips complexes form a filtration, and that is why homologies of different dimensions can be computed. They form a filtration, so that the following property holds:

$$R(Z, \varepsilon) \subseteq R(Z, \varepsilon') \text{ , whenever } \varepsilon < \varepsilon' \tag{2}$$

This property means that corresponding homologies of complexes also form a filtration, so that:

$$H_k(R(Z, \varepsilon)) \subseteq H_k(R(Z, \varepsilon)) \text{ , whenever } \varepsilon < \varepsilon, \text{ for each } k \tag{3}$$

Using this properties it is possible to find birth and death moments of some α class of l-dimensional homologies. This (birth, death) pairs generate persistence diagram, which shows all persistent homologies of a point cloud. Important note is that for the analysis we focus on 1-dimensional persistent homologies and ignore 0-dimensional, following most papers on the topic.

After we had obtained persistence diagrams (PDs), we need to use them for clustering of the original time series. However, common methods cannot be applied on the space of persistence diagrams, as they have different statistical features, so that they cannot be just added or subtracted, and what is most important, common distance measures do not work on this set. One approach to find the distance between diagrams is to use Wasserstein distances.

Degree-p Wasserstein distance between probability measures μ and ν in $\subset P(\Omega)$ can be calculated as following:

$$W_p(\mu, \nu) \stackrel{\text{def}}{=} \left[\inf_{\pi : \Pi(\mu, \nu)} \int_{\Omega^2} D(x, y)^p d\pi(x, y) \right]^{1/p} \tag{4}$$

So we use that as a measure of distance between PDs. And afterwards one can use Wasserstein barycenters as a measure of means. As suggested by Cuturi and Doucet (2014), a Wasserstein barycenter of N measures $\{\nu_1, ..., \nu_N\}$ in $P \subset P(\Omega)$ is a minimizer of f over P, where

$$f(\mu) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{i=1}^{N} W_p^p(\mu, \nu_i) \tag{5}$$

Afterwards one can apply modified k-means with Wasserstein distances and Wasserstein barycenters instead of Euclidean metrics, as was suggested by Maroulas et al. (2017). However the main feature of our datasets (which share most commercial time series) is that they contain a lot of time series, and they are needed for careful analysis, so the process of computation of distances and barycenters in this case is computationally intense, which can become a great problem. Another problem of this algorithm is that it is not theoretically proved and actually on the real data it does not converge, as we revealed, which will be more precisely discussed in section 4

As a possible solution of this problem the approach was suggested by Lacombe et al. (2018) in his work on "Large Scale computation of Means and Clusters for Persistence Diagrams using Optimal Transport". The main idea of his approach is that the optimal transport is approximated with iterative approach.

In the original paper optimal transport is defined as follows: "Given a space χ with a cost function $c : \chi \times \chi \to R_+$, we consider two *discrete* measures μ and ν on $'chi$, namely measures that can be written as positive combinations of diracs, $\mu = \sum_{i=1}^{m} a_i \delta_{x_i}$, $\nu = \sum_{i=1}^{n} a_i \delta_{y_i}$, with weight vectors $a \in R_+^n$, $b \in R_+^m$, satisfying $\sum_i a_i = \sum_i b_i$ and all x_i, y_i in χ. The $n \times m$ cost matrix $C = (c(x_i, y_i))_{ij}$ and the transportation polytope $\Pi(a, b) := \{P \in R_+^{n \times m} | P\mathbf{1}_m = a, P^\top \mathbf{1}_n = b\}$, define an optimal transport problem whose optimum L_C can be computed using either of two linear programs, dual to each other:

$$\mathbf{L}_C(\mu, \nu) := \min_{P \in \Pi(a,b)} \langle P, C \rangle = \max_{(\alpha,\beta) \in \Psi_C} \langle \alpha, a \rangle + \beta, b \rangle \tag{6}$$

where $\langle \cdot, \cdot \rangle$ is the Frobenius dot product and Ψ_C is the set of pairs of vectors (α, β) in $R^n \times R^m$ such that their tensor sum $\alpha \oplus \beta$ is smaller than C"

In order to solve the problem of optimal transport for large data Cuturi and Doucet (2014) proposed the requlatized form of the problem with entropy:

$$\mathbf{L}_C^\gamma(a, b) := \min_{P \in \Pi(a,b)} \langle P, C \rangle - \gamma h(P) \tag{7}$$

where $\gamma > 0$ and $h(P) := -\sum_{ij} P_{ij}(\log P_{ij} - 1)$. Therefore the solution takes the following form:

$$P^\gamma = diag(u^\gamma) K diag(v^\gamma) \in R^{n \times m} \tag{8}$$

where $K = e^{-\frac{C}{\gamma}}$, obtained using term-wise exponentiation, and $(u^\gamma, v^\gamma) \in R^n \times R^m$ is a fixed point of the Sinkhorn map:

$$\mathbf{S} \ : \ (u, v) \to \left(\frac{a}{Kv}, \frac{b}{K^\top u} \right) \tag{9}$$

where all operations are done term-wise. After that Cuturi and Doucet (2014) suggests to consider "the transport cost of the optimal regularized plan, $\mathbf{S}_C^\gamma(a,b) := \langle P^\gamma, C\rangle = (u^\gamma)^\top (K \odot C)v^\gamma$ to define a Sinkhorn divergence between a and b. Therefore as the main result we have that $\mathbf{S}_C^\gamma(a,b) \to \mathbf{L}_C^\gamma(a,b)$ as $\gamma \to 0$". This approximation is further applied to the approximation of distances and barycenters.

Further Lacombe suggests that we need to discretize our persistent diagrams for more convenient work. We encode them as histograms on the square, so we can represent them as $d \times d$ square matrices. The encoding process was as follows:

- We calculate the maximum (the most persistent) point among all PDs in given dataset
- We therefore create square matrix $\mathbf{a} \in R^{d\times d}$, for which each element a_{ij} corresponds to the number of points in original PD, located in cell at position $(i,\ j)$ on the grid
- We then associate to each histogram its mass, which corresponds to the number of off-diagonal points in the original PD, and this fact is extremely important feature of this approach

For further analysis we use $(\mathbf{u}, u_\Delta)$ where $\mathbf{u} \in R^{d\times d}$ is the matrix representation of histogram, and $u_\Delta \in R_+$ is its corresponding mass.

Following the approach of Lacombe, we define cost matrix $C \in R^{(d^2+1)\times(d^2+1)}$ and kernel matrix K as follows:

$$C = \begin{pmatrix} \hat{C} & \vec{\mathbf{c}}_\Delta \\ \vec{\mathbf{c}}_\Delta^\top & 0 \end{pmatrix}, \quad K = \begin{pmatrix} \hat{K} := e^{-\frac{\hat{C}}{\gamma}} & \vec{\mathbf{k}}_\Delta := e^{-\frac{\vec{\mathbf{c}}_\Delta}{\gamma}} \\ \vec{\mathbf{k}}_\Delta^\top & 1 \end{pmatrix} \tag{10}$$

where: $\hat{C} = (\|(i,i') - (j,j')\|_p^p)_{ii',jj'}$, $\mathbf{c}_\Delta = (\|(i,i') - \pi_\Delta((i,i'))\|_p^p)_{ii'}$ (here π_Δ is the projector on the diagonal)

However, we can apply matrix convolutions for the matrix multiplication $K\mathbf{u}$, which is used in 9. For this we do not need to calculate matrices C and K, but rather can rely on $\mathbf{c}_\Delta$, $\mathbf{k}_\Delta = e^{-\frac{\mathbf{c}_\Delta}{\gamma}}$, matrix $\mathbf{c}$, which consists of elements $\mathbf{c}_{ij} = |i-j|^p$ and matrix $\mathbf{k} = e^{-\frac{\mathbf{c}}{\gamma}}$.

Using this matrices and by application of matrix convolution we arrive to the following formula for the calculation of K applied to $(\mathbf{u}, u_\Delta)$:

$$(\mathbf{u}, u_\Delta) \to \left(\mathbf{k}(\mathbf{k}\mathbf{u}^\top)^\top + u_\Delta \mathbf{k}_\Delta, \langle \mathbf{u}, \mathbf{k}_\Delta\rangle + u_\Delta\right) \tag{11}$$

where $\langle\cdot,\cdot\rangle$ is the Frobenius dot product.

Furthermore, in order to calculate the approximation of optimal transport itself, which is $\mathbf{S}_C^\gamma$, we introduce $\mathbf{m} := \mathbf{k} \odot \mathbf{c}$ and $\mathbf{m}_\Delta := \mathbf{k}_\Delta \odot \mathbf{c}_\Delta$. Thus the transport cost of $P = diag(\vec{u}, u_\Delta)K diag(\vec{v}, v_\Delta)$ can be computed as follows:

$$\langle diag(\vec{u}, u_\Delta)K diag(\vec{v}, v_\Delta), \hat{C}\rangle = \langle diag(\vec{u})\hat{K} diag(\vec{v}), \hat{C}\rangle + u_\Delta\langle \mathbf{v}, \mathbf{m}_\Delta\rangle + v_\Delta\langle \mathbf{u}, \mathbf{m}_\Delta\rangle \tag{12}$$

where $\langle diag(\vec{u})\hat{K}diag(\vec{v}), \hat{C}\rangle + u_{\Delta}\langle \mathbf{v}, \mathbf{m}_{\Delta}\rangle = \|\mathbf{u} \odot (\mathbf{m}(\mathbf{k}\mathbf{v}^{\top})^{\top} + \mathbf{k}(\mathbf{m}\mathbf{v}^{\top})^{\top})\|_1$

Another function which is used in the further calculations is $\mathbf{R}$ which is a linear operator which associates a square matrix of histogram with its mass. All the functions, defined above, are needed for computation of the approximated distances by Algorithm 1 and for computation of barycenters by Algorithm 2, which will be more precisely discussed in section 4.

3 Data description

The main goal of our project is to use TDA for clustering of commercial time series, for which we use two datasets, taken from kaggle competitions Elo Merchant Category Recommendation and Predicting Red Hat Business Value. Both datasets share the same structure: for each user they have time series of activities on the given day. We preprocessed the data as follows:

1. We have taken 10k random users as a sample from original dataset, in order to be able to check our approach in the finite time.

2. We dropped all unnecessary features and categories, saving just ids, timestamps and activities.

3. We created the time series of equal sizes from the original time series, which depicted the number of activities per day, made by each of the users.

On the figures below you can see several time series for randomly chosen users among both datasets.

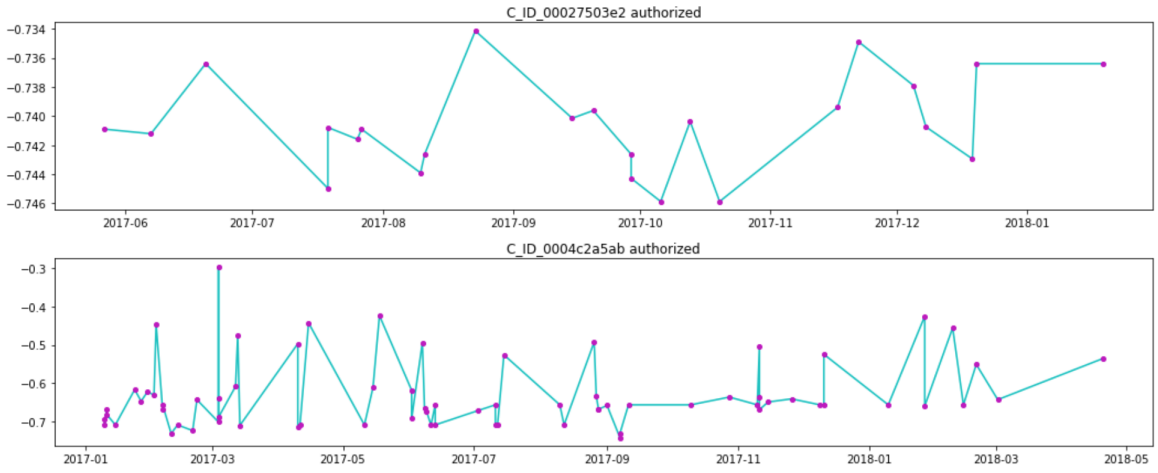


Figure 1: Time series from Elo Merchant data set

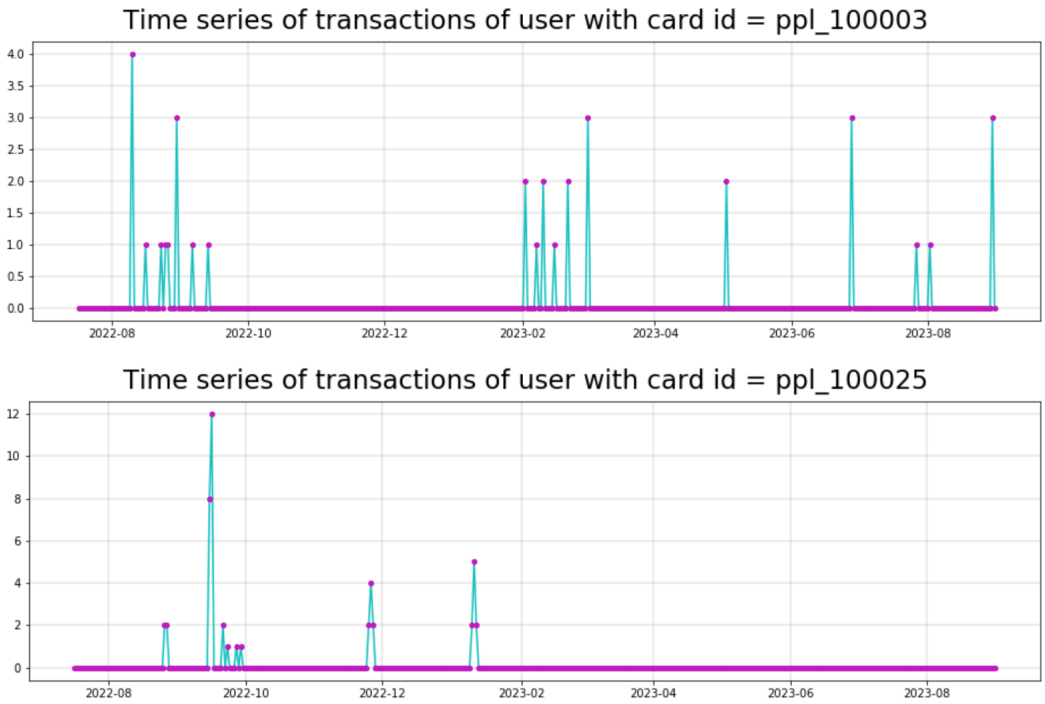


Figure 2: Time series from Red Hat data set

After that we create for each dataset the set of persistence diagrams:

4. We apply sliding window embedding to time series in order to create point clouds.

5. We then generate persistence diagrams for each point cloud, using the Rips filtrations

The illustration of this pipeline you can find on the figure 3 which represents the process for one random user from Elo Merchant dataset.

This process was previously done by one of the participant of our team, therefore we have provided the code for such creation and in the notebook we use already prepared data set with persistence diagrams of each time series for both datasets, which we then use in our work. However, during the analysis it turned out that the red hat dataset contains PDs with very little points, which are mostly located on the diagonal, so we do not use it in our analysis.

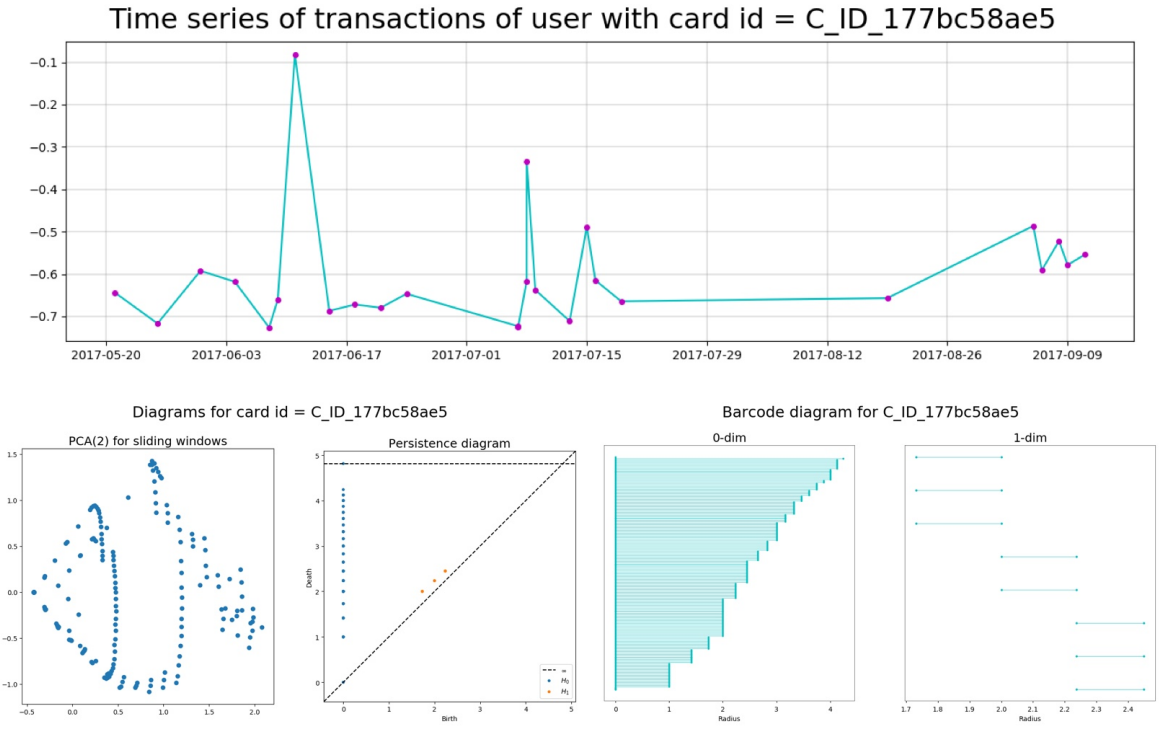


Figure 3: Pipeline for the creation of persistence diagrams

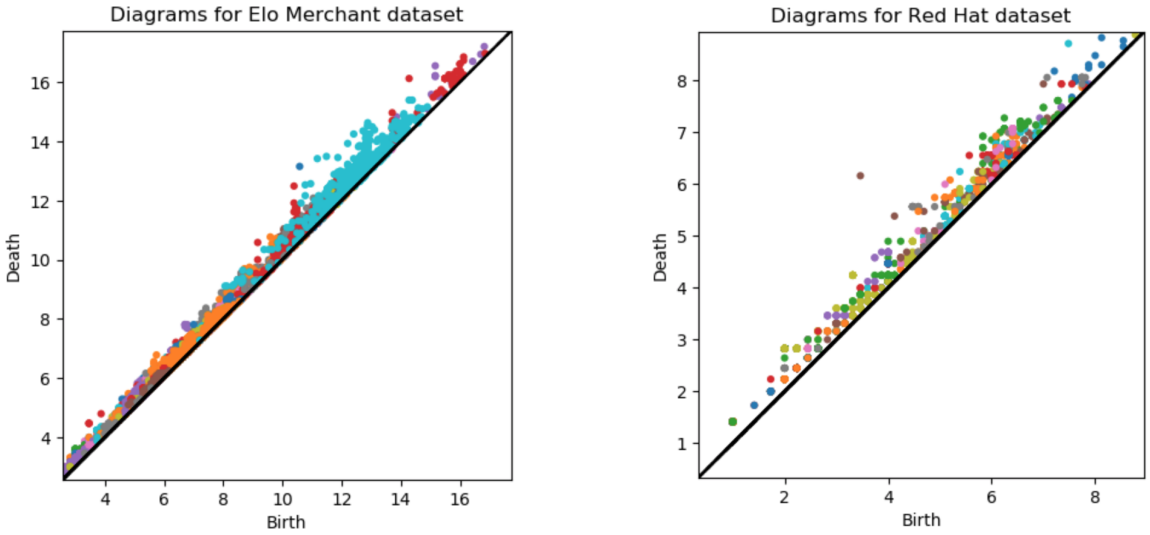


Figure 4: Persistence Diagram.

In order to check our approach we also use synthetic persistence diagrams, which we create using `make_blobs`. We create blobs of different sizes, which are located above the diagonal, and we use them for testing our algorithms. Several examples of such blobs can be found on the figure 5.

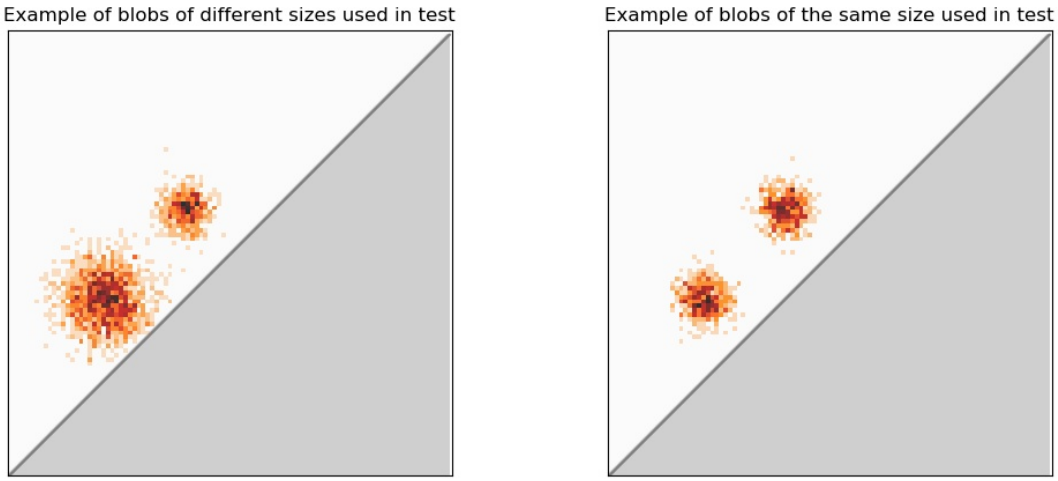


Figure 5: Example of blobs

4 Methodology

It is very important to note, that original algorithms, if recreated as followed directly from the paper, does not work, because several important clarifications are needed. We first implemented the original algorithms and they did not converge, but after we had a conversation with Theo Lacombe we accounted for his clarifications, which were not explicitly stated in the paper, and thus implemented slightly changed algorithm. Here we present implemented by us algorithms with the changes, that the author suggested. Among the important clarifications are specifically defined initial states (u, v), the thresholding of the elements of k and the definition of operator $R^\top$

The first algorithm (referred to as Sinkhorn divergence for persistence diagrams in the paper) is aimed to approximate distance between pair of diagrams. In the beginning we have to initialize u and v, used in the Sinkhorn step from 9, as uniform distributed objects from $\mathrm{R}^{\mathrm{d}\times\mathrm{d}} \cup \{\Delta\}$. This step was skipped in the paper, but suggested to us directly by an author. Also we have to initialize γ - smoothing parameter, d - grid step and p — order of norms, used in computations. Another additional parameter is threshold for elements of matrices, which ensures numerical stability of his algorithm by substituting too small elements by just very small elements. This algorithm is described below:

Algorithm 1 Sinkhorn divergence for persistence diagrams

Input: Pair of PDs, encoded as histograms $\mathbf{a}$ and $\mathbf{b}$, smoothing parameter γ, grid size d, order of norm p, threshold for small values
Output: Approximation of distance between two PDs
initialize u_0 and v_0 as uniform measures. We initialized them as matrices, filled with $\frac{1}{d^2}$. Also we create kernel matrices for given d and p
repeat for selected number of iterations
Iterate the Sinkhorn step, defined in 9, using the convolution from 11
Compute $\mathbf{S}_{\mathbf{C}}^{\gamma}(\mathbf{a}+\mathbf{Rb}, \mathbf{b}+\mathbf{Ra})$ using the formula 12

For computation of barycenters, we apply algorithm 2 (referred to as Smoothed approximation of PD barycenter in the paper), which is the case of mirror descent, as suggested by the author. For the implementation of this algorithm another clarification from the author was needed, regarding the linear operator $R^{\top}$, which was not defined in paper, but turned out to the matrix filled with the mass of the histogram, to which it is applied. Therefore the second algorithm can be summarized as follows:

Algorithm 2 Smoothed approximation of barycenters

Input: List of PDs, encoded as histograms, learning rate λ, smoothing parameter γ, grid size d, order of norm p, threshold for small values
Output: Approximation barycenter $\mathbf{z}$
initialize z as uniform measures. We initialize is as matrix, filled with $\frac{1}{d^2}$ above diagonal. We also create kernel matrices
repeat until z changes
Iterate the Sinkhorn step, defined in 9, between all the pairs $(\mathbf{z}+\mathbf{Ra}_i)_i$ and $(\mathbf{a}_i+\mathbf{Rz})_i$, using the convolution from 11
Compute $\nabla := \gamma\left(\sum_i \log(u_i^{\gamma}) + R^{\top}\log(v_i^{\gamma})\right)$
$\mathbf{z} := \mathbf{z} \odot exp(-\lambda\nabla)$

Also we decided to implement algorithm for computation of Wasserstein Barycenters computation, as defined in 5. Our goal was to compute them and compare this algorithm somehow with algorithm of Theo Lacombe. We implemented this iterative method as following: we first define kernel K, in this case kernel associated with squared Euclidian norm is a convolution with a Gaussian filter. That is why multiplication with kernel can be computed efficiently, using convolution methods. Our next step was to define weights for isobarcenters. For this reason we used Sinkhorn-like algorithm $P_k = diag(u_k)K(\nu_k)$, for some positive weights u_k, ν_k.

After that we use Bregman projection method, on the first step this correspond to the projection on the fixed marginals constrains, this achieved by updating: $\forall k = 1, \ldots, R, \quad u_k \longleftarrow \frac{a_k}{K(v_k)}$. On the second step Bregman projection correspond to the projection on the fixed marginals constrains for common barycenter.

In the end, we include this algorithms in our own k-means clustering. We find centroids as barycenters for sets of diagrams, and after that we approximate Wasserstein distances

between diagrams and barycenters, like in the normal k-means. But this approach does not converge. One of the reasons for this is that Wasserstein distances is not suitable in case of optimal transport problem. Our experiments prove that this is true, as k-means with such centroids and distances does not converge.

5 Results

First of all, we checked how well distances are approximated and whether they make sense at all. First of all we made sure that the distance between similar diagrams is indeed zero, while for different diagrams it increases. In order to check this mechanism even further, we have created two tests: one with two similar blobs, and another with two blobs with different masses, in order to see, how algorithm reacts on the unequal masses. As the result we revealed that algorithm works well and the results of it are reasonable. On the figure 6 you can find the illustration of tests' results

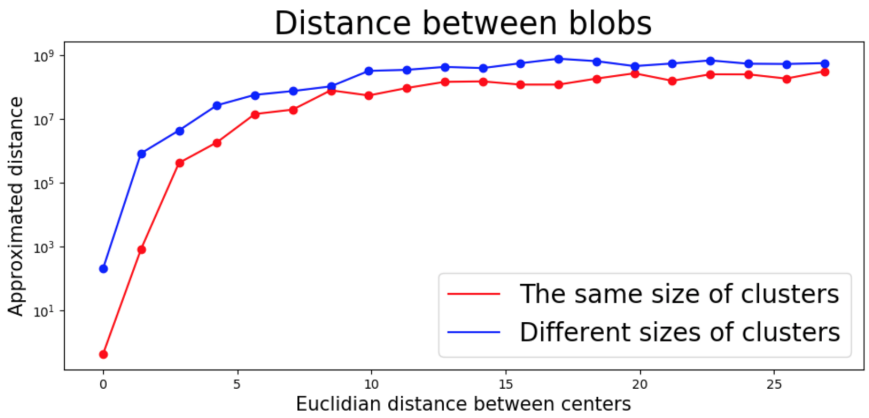
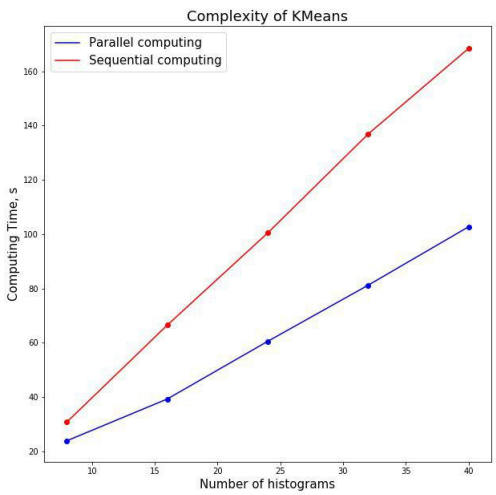


Figure 6: Distances between different blobs

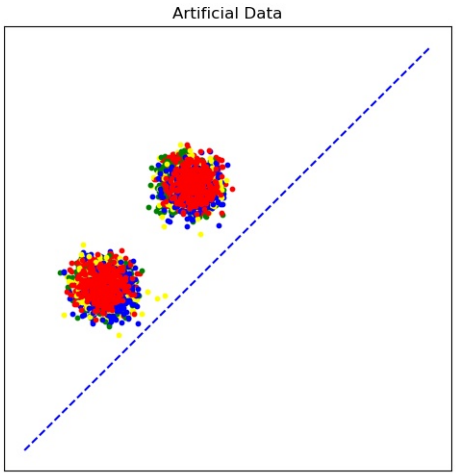
We have created our own k-means, which included approximated distances and computation of smoothed barycenters inside of it. We have tested our algorithm on several sets of artificial data, and it indeed converged in case of two blobs (for each blob we had several PDs with points inside that blob), but we had several computational ambiguities in case of 4 blobs. Moreover, Theo Lacombe in conversation mentioned that they had also implemented very naive k-means clustering and he did not have checked it on the variety of datasets. Therefore we conclude that this algorithm, while it generally works well, harshly depends on the initial data. We suggest for future research to include some sort of normalization, which in space of PDs is a tricky thing to do.

Also we parallelized the main computational steps in k-means algorithm: calculation of distances and calculation of barycenters, as was suggested by Theo Lacombe, in order to speed up our clusterization process. The method of parallelization based on multiprocessing. We use joblib library for handmade parallelization of main parts of code and tried numba for automatic parallelization of other part. For this on the figure 7 you can see the compari-

son of execution time of parallelized and sequential algorithms for different types of artificial datasets, as well as an example of artificial datasets, used in tests.



Comparison of execution time



Artificial Data example

Figure 7: Comparison of execution times

Also we have implemented Wasserstein barycenter computation algorithm, which was used for more nice visualization and as some possible alternative, which we have proved to be non-working.

6 Conclusion

This project investigates the possibility to cluster time series with the use of TDA, using the computationally efficient approach, suggested by Lacombe et al. (2018). We have recreated algorithms for approximation of optimal transport and barycenters, which indeed work, but with several important clarifications of the algorithm. Additionally, we have recreated another approach, based on the ideas to use Wasserstein distances and Wasserstein barycenters for k-means clustering of PDs.

As the result, we have revealed that the algorithm 1 indeed works on the real data, as well as it works on artificially created blobs. We have also checked that algorithm indeed incorporates the mass in the approximation of distances. Then we made sure that the algorithm 2 works on the artificial data. Afterwards, we have incorporated them in the algorithm of k-means clustering and it worked well on artificial data, while had problems with convergence on the real dataset.

Therefore for the future research we suggest that the normalization of data is needed, which is a tricky task. Also we suggest to improve the parallelization mechanism of the k-means, as

well as inclusion of some other clustering algorithm.

References

L. M. Seversky, S. Davis, and M. Berger, "On Time-Series Topological Data Analysis: New Data and Opportunities," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1014–1022, 2016.

M. Gidea, D. Goldsmith, Y. Katz, P. Roldan, and Y. Shmalo, "Topological recognition of critical transitions in time series of cryptocurrencies," pp. 1–29, 2018. [Online]. Available: http://arxiv.org/abs/1809.00695

Y. Umeda, "Time Series Classification via Topological Data Analysis," Tech. Rep. 3, 2017. [Online]. Available: https://www.jstage.jst.go.jp/article/tjsai/32/3/32{_}D-G72/{_}pdf

T. Lacombe, M. Cuturi, and S. Oudot, "Large Scale computation of Means and Clusters for Persistence Diagrams using Optimal Transport," Tech. Rep., 2018. [Online]. Available: http://arxiv.org/abs/1805.08331

M. Cuturi and A. Doucet, "Fast Computation of Wasserstein Barycenters," Tech. Rep., 2014. [Online]. Available: https://arxiv.org/pdf/1310.4375.pdf

V. Maroulas, J. Mike, and A. Marchese, "Kmeans clustering on the space of persistence diagrams," in *Wavelets and Sparsity XVII*, Y. M. Lu, M. Papadakis, and D. Van De Ville, Eds., vol. 10394. SPIE, aug 2017, p. 29. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10394/2273067/Kmeans-clustering-on-the-space-of-persistence-diagrams/10.1117/12.2273067.full