TDA

Topological Data Analysis - group project

Bernarda Petek, Jon Selič

1 Introduction

Topological data analysis (TDA) is a modern tool for data analysis which can be used in many different ways. Even though the community is generally agreed on a standard TDA pipeline, many decisions still have to be made by the analyst which can greatly affect the results and their interpretation. In this project we applied five different variations of the TDA pipeline and compare the results.

2 Data and Methods

2.1 Obtaining Pointclouds

Data was obtained from two different sources, Our World in Data[1] and Data World Bank[2]. For each country with at least one year of data we obtained GDP per person employed, population, refugee population, percentage of people enrolled in primary school, and total number of working people throughout the years 1960-2021. We preprocessed the data by removing the rows with missing parts of data. We also filtered the data by removing the countries which had less than 10 years worth of data.

So for each country we got at least 10 years worth of data which consisted of GDP per person employed, population, refugee population, percentage of people enrolled in primary school, and total number of working people. Then we standarized the data. Finally, to get pointclouds, we partitioned the data by countries.

2.2 Filtration and Computing Persistent Homology

Next, we computed persistent homology for each pointcloud and represented it with persistence diagram. But to do that we first had to build a filtration on a simplicial complex constructed by points in a pointcloud. For building Vietoris-Rips filtration and computing persistent homology we used Ripser.py library. For each pointcloud we computed 0 and 1 dimensional persistent homology groups because higher dimensions are usually harder to compute. Before computing persistent homology on our datclouds we computed persistent homology on a dummy data. We made a few pointclouds where every point had four random

[1] url
[2] url

coordinates ranging from 1 to 100. We computed persistent homology up to dimension 3. Computing persistent homology for dimensions 2 and 3 indeed took a long time. However, we observed that random dummy data had some one- and even two- diensional holes. Then when we computed persistent homology on some of our data we realized our data is usually shaped in a way that there are no holes. Subsequently we realized that in our case we only needed to compute 0 dimensional persistent homology and its persistence diagram. That made sense because our data was linearly changing through the time. So we computed 0 dimensional persistent homology and represented it with persistence diagram for each pointcloud.

2.3 Space of Persistence Diagrams and Clustering

After computing 0-dimensional persistent homology for every pointcloud, we got a space of persistence diagrams. We computed two different distance matrices for two different metrics on the space of persistence diagrams. For the first distance matrix we used bottleneck distance between persistence diagrams and for the second distance matrix we used Wasserstein distance between persistence diagrams. After computing distance matrices we used agglomerative clustering algorithm on the space of the persistence diagrams to try to find similarities between our pointclouds. We linked persistence diagrams which were put together by the clustering algorithm to the original pointclouds. We got two different dendrograms which showed us the similarities and differences between the shape of the data with the help of the persistence diagrams.

2.4 Vectorization and Clustering

In the next step, we extended our pipeline by vectorizing the data from the persistence diagrams before using it in a clustering algorithm. We did this in two ways: Firstly, by computing the component weights for the Gaussian mixture model (GMM) that has been trained on all of our diagrams and computing the distance between them - and secondly, transforming the persistence diagram into a pixelated persistence image and computing the distance between each pixel in the transformed image. The distance between two images was computed in two different ways: A simple euclidean distance metric, taking the flattened images as input vectors, and the average mean squared error (MSE) between the pixels of the two images. The resulting two distance matrices were then used in the clustering algorithm, similarly as in the previous step. The result was three additional dendrograms.

3 Results

4 Conclusion

5 Division of Work