FGMDA - Pedestrian Recognition

Clément Piat - Alexandre Joutard

March 2021

1 Introduction

The goal of this is project to illustrate, on a toy example, the benefit of coordinate invariance of persistent homology. The walks of 3 pedestrians A, B and C have been recorded using the accelerometer sensors of smartphones carried in their pockets, leading to 3 multivariate time series in $\mathbb{R}^3$: each time series represents the 3 coordinates of the acceleration of the corresponding pedestrian in a coordinate system attached to the sensor. Since smartphones were carried in unknown different positions and were not fixed, there is no correspondence between the coordinate systems of each smartphone, and thus these time series cannot be compared using coordinates.

Using a sliding window, each time series has been split in a list of **100 times series made of 200 consecutive points**, that are stored in data A, data B and data C. To each set of 200 points is associated a label A, B or C. This project aims at doing supervised learning with persistence diagrams in order to achieve pedestrian recognition.

2 Code

We used Python to develop the solution. Part of the code is written in raw .py files, part of it is written in a jupyter notebook. Everything can be found on Github (`https://github.com/clementpiat/pedestrians_recognition`).

3 The data

As we said in the introduction, we have 300 labeled samples, each of them is a (200x3) vector representing the temporal evolution of 3D coordinates, and we are trying to classify them in 3 classes A, B, and C.

4 Persistent diagrams

We leverage the gudhi library to compute persistent diagrams of every sample (i.e. every "3D time series"). To compute these diagrams we must choose the

dimension of the simplices we consider, and the maximum length for 2 3D points to be connected in the construction of the persistent diagram.

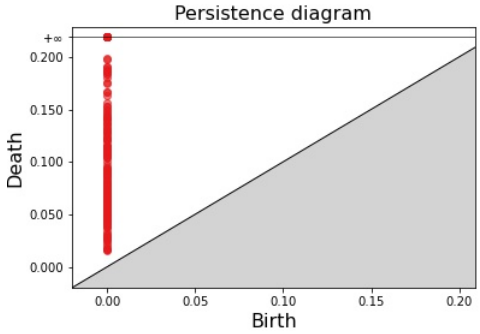

Figure 1: Persistent diagram example in dimension 0 with $max\_length = 0.2$

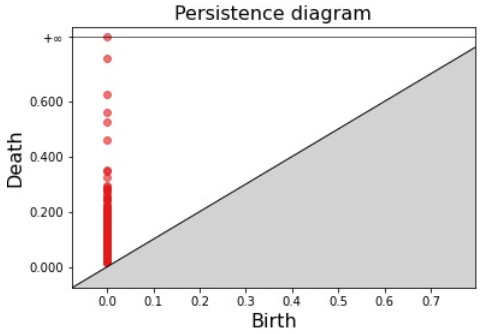

Figure 2: Persistent diagram example in dimension 0 with $max\_length = 10$

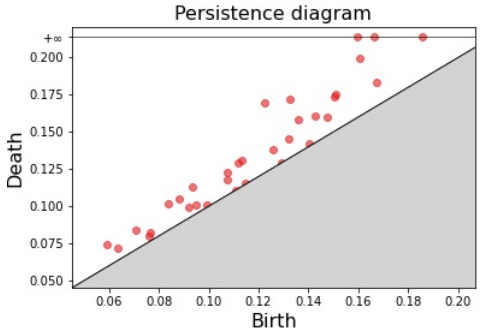

Figure 3: Persistent diagram example in dimension 1 with $max\_length = 0.2$

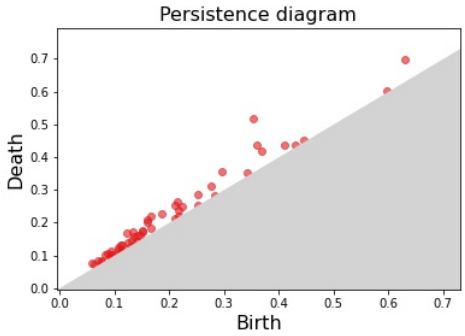

Figure 4: Persistent diagram example in dimension 1 with $max\_length = 10$

We can see that in dimension 0 there is only one sample with an infinite value for a maximum length of 10. This is because for a big value of the maximum length, we have only one connected component at the end.

Also we can notice that in dimension 0, every point in the diagram has a birth value of zero, which is coherent because a simplex is a point in 0d, not an edge like in 1d.

The $300 \times 2 = 600$ persistent diagrams were computed almost instantly.

5 Distance matrices

We computed the distance matrices by leveraging again the gudhi library and the bottleneck distance they implemented for computing a distance between 2 diagrams. Then we used the sklearn library to project the samples in 2d using the MDS algorithm and the distance matrix.

We used the persistent diagrams computed with a maximum length of 10, because we wanted our diagrams to have the same number of infinite values, so that the distance between them is finite and we can use MDS. The value of 10 was chosen empirically.

This took a little bit more time, something like 10 minutes to compute the 2 distance matrices (one for each dimension).

6 Persistent landscapes

Then, as suggested, we computed the persistence landscapes from each persistence diagram that we had computed (0 and 1 dimension). For this, we used again the Gudhi library and its function Landscape. It has several arguments which are the number of landscape to consider, the number of points and a range of coordinates to represent each landscape. Thus, we built a function which given the list of all the persistence diagrams will output a vector representing the landscapes.

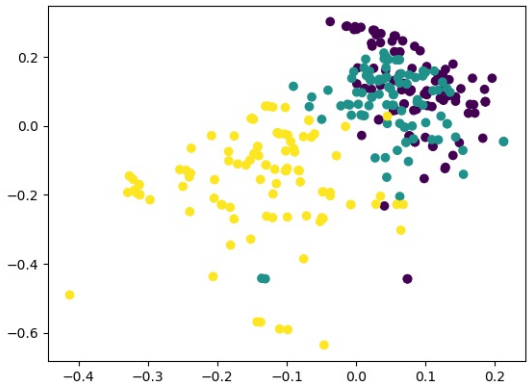

Figure 5: Projections of the 0D persistent diagrams after MDS

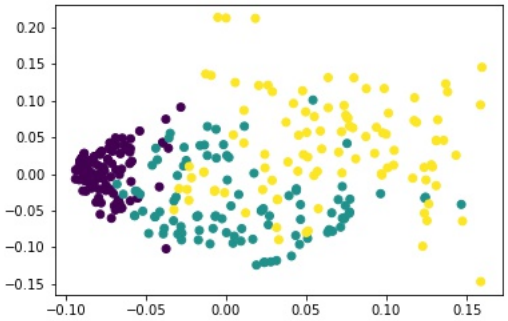

Figure 6: Projections of the 1D persistent diagrams after MDS

7 Classification

Finally, we used the landscapes to classify the pedestrians. Instead of feeding the raw data to a classifier, the landscapes could be much better features, and could lead to better results. Thus, using a RandomForest algorithm as a classifier, we compared the use of raw data and the landscapes. We splited the data in a train set and validation one with a ratio of 0.8. We mainly optimized the RandomForest with the n_estimators parameter. We chose accuracy as metric since the data set is balanced.

7.1 Classification with landscapes

We decided to compute 5 landscapes for each persistence diagram and we tested several number of points to sample these landscapes, around 100. Finally, we fixed the range of the landscapes coordinates to $[0, 0.8]$. So we had approximately 500 points per sample instead of 300. And we could consider both the 0-dim and 1-dim persistence diagrams, which will lead to 1000 features.

After computing the different landscapes, we achieved 100% of accuracy using only 1-dim diagrams. Surprisingly, the results were worse when we tried to use both 0-dim and 1-dim diagrams. The results of 0-dim diagrams were not so great but we could have played more on the hyper parameters probably to improve. On the second graph, we searched for the better number of nodes to represent the landscapes and and the number of trees in the RandomForest and achieved 98.3% with both dim.

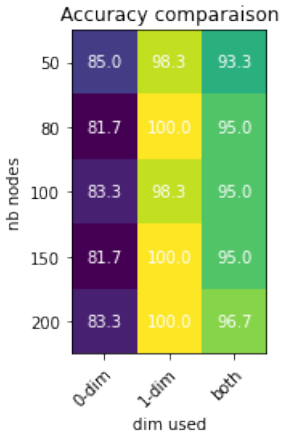

Figure 7: Accuracy comparison of landscapes using different diagrams and number of nodes for each landscape

7.2 Classification with raw data

Then we compared our results with the use of raw data. Thus we trained the RandomForest classifier using only the 300 points representing each time lapse and tried to optimize the hyper parameter of the classifier. For this, we played on the number of trees again and also the parameter min_samples_split which is the minimum number of samples per leaf (since we have very few data, it shouldn't be too much). We get good results here also with 98.3 % accuracy on the validation set. But we didn't succeed to get 100% contrary to landscapes' features.

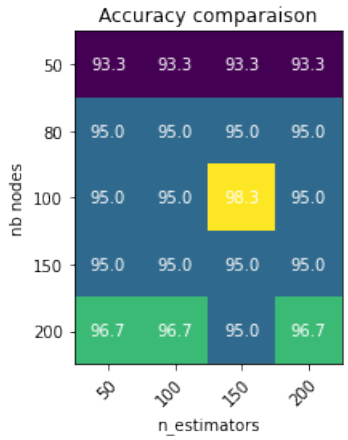

Figure 8: Accuracy comparison of landscapes using both diagrams

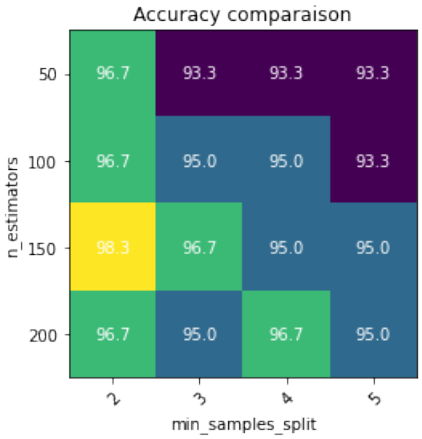

Figure 9: Hyper parameters optimization with raw data

8 Conclusion

The main takeaway from this project, and this course actually, is that topological data analysis and persistent homology is very useful in the sense that it provides us with new tools to infer descriptors from geometric data, like the persistent diagrams or the persistent landscapes as illustrated in this exercise.

Persistent diagrams and persistent landscapes are very visual, but also proved to be very meaningful descriptors of the data in this particular usecase of pedestrian classification.