TDA pipeline

Matic Stare Martin Starič

January 3, 2024

Contents

Abstract

In this report we explore the capabilities of the TDA pipeline to classify the number of clusters within a point cloud by training a classification model. We generate synthetic data, create a filtration, build persistence diagrams and map them to an Euclidean space using vectorization techniques. We then use statistical methods to interpret the data and train a classification model. Furthermore, we test our model on the Iris dataset and different synthetic data.

1 Introduction

The standard TDA pipeline includes creating a filtration, building persistent diagrams and mapping them to an Euclidean space using vectorization techniques, then use statistical methods to interpret the data. In this report we explore the capabilities of the TDA pipeline to classify the number of clusters within a point cloud by training a classification model.

2 Methods

2.1 Synthetic training data

To create a classification model, we require a set of data to train it on. In order to create a large enough set of data, we used an algorithm that generates a cloud of points, such that there are between $100 - 500$ points and $2 - 6$ clusters in a 3D space. We generated 1000 different instances of point clouds, where the first 200 point clouds represent data with two clusters, next 200 represent data with three clusters, and so on up to six clusters.

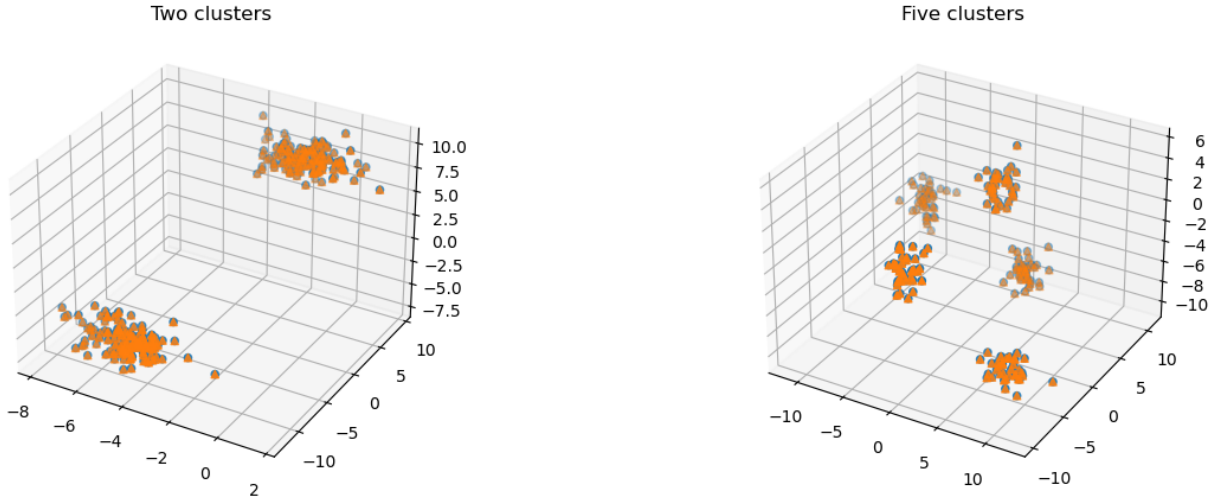

Figure 1: Training set examples, left a point cloud with two clusters, right a point cloud with 5 clusters

Above we see a figure of our synthetic data, the left plot represents a point cloud that forms two clusters, while on the right plot a point cloud that forms five clusters.

2.2 Vietoris-Rips and Persistence diagrams

Vietoris-Rips is our choice of filtration, as it is simple. Simplicity is key for performance goals, especially since we are only required to make persistence diagrams of dimension 0. To distinguish the number of clusters in a point cloud, we measure the number of components remaining after a certain time stamp. The majority of points would connect in the earlier parts of the filtration, whereas clusters would only connect after a much later time, that is the clusters are far enough apart. This is the reason why only dimension 0 is relevant.

Remark - We have attempted to use persistence diagrams with dimension 1 as well, however this captures features that actually worsen our classification model, so we decided to only use dimension 0.

2.3 Persistent images

To vectorize the obtained persistence diagrams, we chose persistent images. The algorithm typically creates a 2D image from each persistence diagram which we would proceed to flatten. However, since our persistence diagrams are only of dimension 0 and our birth coordinate for each component is fixed to 0, we directly obtain a one dimensional image from this vectorization. Thus, we obtain images of resolution $1 \cdot 100$, which is actually our output vector with 100 features. We chose the number of features to be 100, so that the algorithm would better distinguish when the majority of points die and when clusters start to connect.

As there is no Python implementation of persistent images for H0, we resorted to use the source code from the original authors [1] that made the algorithm in Matlab.

2.4 Creating a classification model

Once the persistent diagrams are vectorized, they are prepared for further analysis. The obtained vectors are normalized, then applied through principal component analysis (PCA), so that the number of features reduces enough to visually interpret the data and preprocess it for training purposes. Then we used Support Vector Machine (SVM) to train a classification model. We chose SVM because it is a powerful tool for classification, and it is also very fast. To ensure that our model is not overfitting, we used cross-validation to test the accuracy of our model. We used a 5-fold cross-validation, which means that we split our data into 5 parts, and each time we train our model on 4 parts and test it on the remaining part. We then take the average of the accuracy of each iteration to get the final accuracy of our model.

3 Results

The results of our classification model are slightly worse than expected. Our accuracy on the training set is 70%. The model has poor accuracy when trying to predict data with 5 clusters. We can see this from the figure below, as it is evident that the classes with four, five and six clusters overlap over each other. This makes that area very difficult to classify for our model.

We tested our model on the Iris dataset, which does not have distinct clusters, but has three classes (Iris setosa, Iris versicolor and Iris virginica) which is what we tried to classify. Our model classified the data incorrectly, it predicted that there were two clusters. From the graph below we can conclude that the Iris dataset is simply too different from the training set for it to be classified correctly.

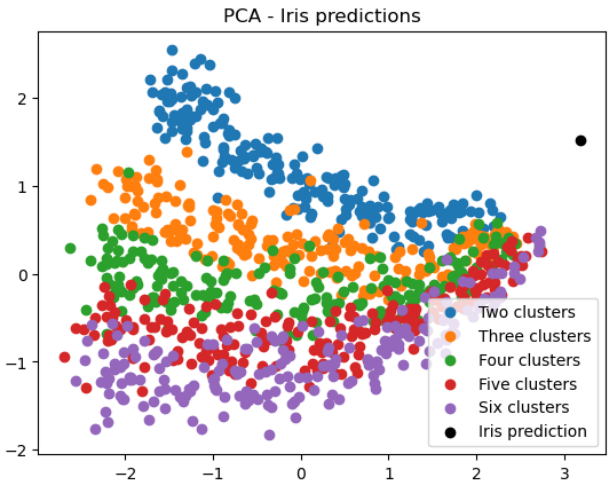

Figure 2: PCA - Iris classification vector is not close enough to data, but closest to blue class

We tried to test our model on different synthetic data, this time two circles in three-dimensional space. The two circles would represent two clusters, so this was our classification goal, which ended up being the case. As shown from the graph below, the vectorized persistence diagram for the two circles data is close enough to the training set and closest to the class corresponding to two clusters.

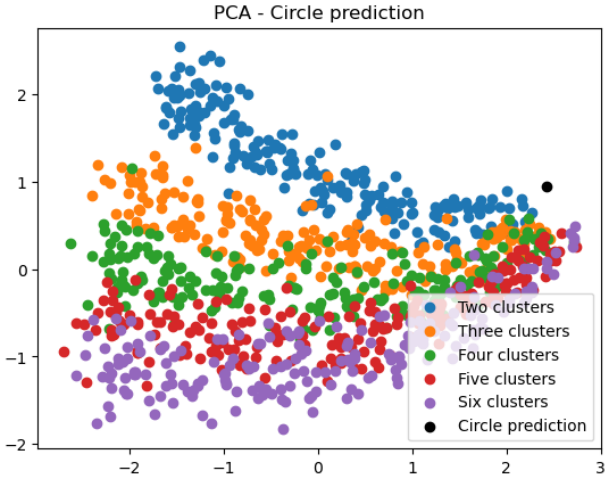

Figure 3: PCA - Circles classification vector is closest to the blue class

4 Discussion

In this report we have explored the possibility of classifying the number of clusters in data with the help of existing tools. The results are not as good as we expected, perhaps because of the choice of vectorization or incorrect parameter choices, as we observed that the persistent images we obtained, are very similar to each other. The weight function in the vectorization emphasizes the components that persist the longest. This way the points that connect within the clusters don't impact the vectorization too heavily. However, neither do the connections of clusters. This is where the problem emerges, since this is already relevant information to our classifier. We could resolve this issue by tuning parameters appropriately, however we were unable to achieve this so far. Another option is to take a different vectorization technique or a wider, more varied set of training data.

5 Division of work

Both authors worked on the entirety of the project together. We took the Extreme programming approach, where we were both working on the same code simultaneously (coding in pairs).

References

[1] Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(8):1–35, 2017.