EN.625.687 Applied Topology HW 8

Lauren Kimpel

11 April 2023

1 Overview

This is a brief walkthrough of my submission. The code (which can be found at `https://github.com/le-kimpel/course_scripts/tree/dev/AppliedTopology` was written in Python 3, and uses a slightly-improved version of the simplicial complex constructed for the previous coding homework.

The code uses the following:

1. `simplicial_complex.py`
2. `TDA.py`

in addition to numerous python libraries, like `pandas, numpy, scipy, sympy`, etc. For each of the .csv files, data is read into a DataFrame, with metric distances between vectors calculated and then shelved into an ordered array. From there, calculations and analyses of persistent homologies are performed. (Or at least, attempted.)

2 The Data

The following .csv files were analyzed:

1. CDHWdata1.csv
2. CDHWdata2.csv
3. CDHWdata3.csv
4. CDHWdata4.csv
5. CDHWdata5.csv

These datasets are "noisy representations of the same thing," with very little to glean from them. Each dataset is a 93×15 dataframe; each 15-dimensional vector contains a variety of floating points. Code was run using an Ubuntu 22.04.1 LTS virtual machine with only about 2 GB RAM.

Here is an example view:

```
cube@cube-vm:~/Desktop/coursework/course_scripts/AppliedTopology$ python3 TDA.py
    partno        x        y         z        u  ...     tngr      u.1        kn        do      it
0        1 -3.03830  2.98430  0.024337 -1.30380  ... -0.86053 -2.37600 -0.026180 -4.000800 -2.9383
1        2 -4.21620  3.06610  0.860380 -1.39590  ... -1.14150 -2.88060 -2.549000 -2.360200 -1.3827
2        3 -4.80700  2.45300  1.070700  0.57314  ... -2.74090 -2.45030 -2.532900 -0.061519 -3.6037
3        4 -3.20220  1.12660  2.834600 -1.32060  ... -0.98405 -1.54620  0.085325 -3.267400 -2.3545
4        5 -2.88230  0.24683  2.685800 -1.73080  ... -1.37940 -3.70610 -1.966400 -4.298600 -2.2616
..     ...      ...      ...       ...      ...  ...      ...      ...       ...       ...     ...
88      89  0.55362  6.59500 -1.643600  0.61403  ...  0.70487  2.23830  2.493800  3.540000  3.8280
89      90 -3.20100  3.87940 -0.394770  0.93890  ...  1.04150  2.60470  1.883600  2.959500  3.1359
90      91 -1.96160  4.62030 -0.330820 -0.86714  ...  1.06490  1.39120  1.676700  3.108000  2.6235
91      92 -1.71790  6.25520  1.201000  1.34180  ...  2.21190  0.86297  3.474800  0.067303  3.2848
92      93 -2.64780  3.51220  1.525800 -0.29238  ...  1.33230  1.62810  2.043500  0.829130  4.1787

[93 rows x 16 columns]
```

Figure 1: A sample of some of our input data (CDHWdata1.csv).

3 Procedure

First, a metric distance is obtained between each of the datapoints. Going with the assignment suggestion, we are to use Euclidean distance. I chose to assign these distances to a 93×93 distance-matrix; from here, each distance gets stuffed into a one-dimensional array and sorted.

I then iterated through as many distances as possible (starting from the smallest possible threshold and working upwards); I went through about 60 iterations. The distance matrix is also instrumental in creating simplices in a "reasonable" timespan.

With each step, the following algorithm is performed:

Algorithm 1 Simple simplicial complex construction, maximum 4 dimensions.

```
for distance in distances do
    sc_data ← FormData(DistanceMatrix, threshold)
    sc_data ← CheckData(sc_data, distance, dimensions = 4)
        SC ← SimplicialComplex(sc_data, dimensions)
            χ ← CalculateEulerCharacteristic(SC)
                X ← distance
                    Y_0, Y_1, Y_2 ← rank SC.H_0, rank SC.H_1, rank SC.H_2

                end for
```

We have to check to make sure we're passing valid data into the simplicial complex structure (e.g., to ensure (1) correct homology calculation and (2) if the structure even counts as a simplicial complex.)

We output both χ, or the Euler characteristic, as well as the ranks of H_0, H_1, and H_2. Note

$$\chi = \text{number of vertices} - \text{number of edges} + \text{number of faces}$$

with these dimensions in mind.

The ranks are then recorded and graphed with `matplotlib.plyplot`. Running the code in this manner, the Euler characteristic χ matched up with the homology ranks.

```python
if __name__ == "__main__":

    df1 = pd.read_csv("Data/CDHWdata_1.csv")
    print(df1)
    D1 = rank_order(df1)
    distances = get_distances(D1)

    # construct the simplices
    for i in range(1,len(distances)):
        C = [get_simplex(D1, 0, distances[i], dim) for dim in range(0,4)]
        #print(C)

        # build the simplicial complex
        Complex = SimplicialComplex(C)

        check_faces(Complex, 4)
        check_faces(Complex, 3)

        H0 = Complex.compute_homologies(1)
        H1 = Complex.compute_homologies(2)
        H2 = Complex.compute_homologies(3)

        print("-------------- S T A T S --------------")
        print("Dimension of SC: " + str(Complex.dimension))
        print("Rank H0: " + str(Complex.compute_homology_rank(1)))
        print("Rank H1: " + str(Complex.compute_homology_rank(2)))
        print("Rank H2: " + str(Complex.compute_homology_rank(3)))
        print("Euler Characteristic: " + str(Complex.compute_euler_characterisic()))
        print("-------------- S T A T S --------------")
```

Figure 2: main code sample.

4 Analysis

Figure 4 depicts a few of the homologies documented in the code. In general, we find that the dimension of H_2 (in the first two .csv files) is always 0, and the dimension of H_1 does not grow nearly as fast as that of H_0.

Judging by the output of the first image, the persistent homologies are $H_1 \approx 5$ and $H_0 \approx 15$. An obvious fact is that, as the radius ρ of Euclidean distances increases, there are more 0 and 1-dimensional holes. H_1 does not pick up a rank greater than zero until we hit $\rho \approx 2.5$, at which point it stepwise increases to 5.

Over time - as we step through each .csv file - H_2 is consistently 0, meaning that there are no 3-dimensional holes in any of these datasets. The maximum number of 1-dimensional holes our dataset has, according to what's been recorded, is 5, though with an even larger radius, this could be inaccurate.

To test this theory, I ran an additional script with `num_iterations` $= 150$. For values of `num_iterations` ≥ 200, the script implodes and is eventually killed by the Linux kernel. Figure 3 contains the output following the conclusion of this test.

While $H_2 = 1$ as ρ increases, H_0 and H_1 spike in different directions. That is, the true number of 1-dimensional holes, as ρ increases, approaches 20, while H_0 drops to 1. It is admittedly difficult to visualize this in terms of connected components.

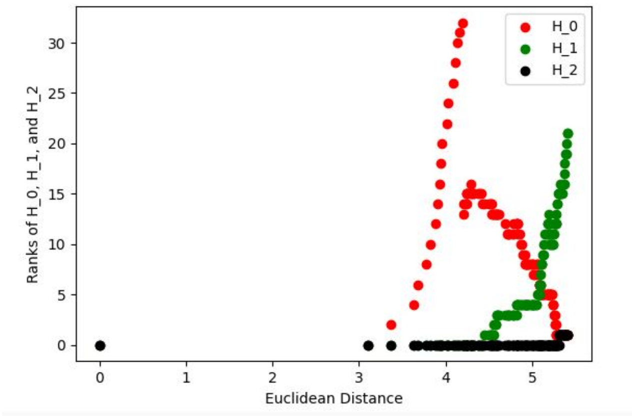

Figure 3: Persistent homologies on noisy datasets 1-5 (left-to-right).

5 Code Weaknesses / Bugs

Some code flops of note are briefly documented here.

5.1 Slow runtimes with large thresholds

I do not particularly opt for sophisticated optimization techniques here. As a result, there are a few functions in the analysis script whose runtimes are *cubic* (not just quadratic) and even $O(n^4)$. Attempting an analysis over a gigantic threshold (such as the entirety of the 93×93-length distance list) is not advised if you're running the same dinky little Ubuntu VM that I am.

5.2 Not generalized to simplices of n dimensions

The simplex-checking employed in the first part of the data analysis script is limited to 4 dimensions, mostly due to time constraints. A nice TODO would be to make this a bit more general. Of course, this would have to be prioritized over *also* making it more optimized, which is a difficult tradeoff.

5.3 Funky χ on large complexes

This isn't necessarily a bug - in fact, it has more to do with having fewer $0-$ simplices than anything else (see this stackexchange post: `https://math.stackexchange.com/questions/3984360/simplicial-complex-and-euler-characteristic`.) For large enough values of ρ, the largeness of H_0 evaporates and we are left with more 1-dimensional holes than 0-dimensional holes.

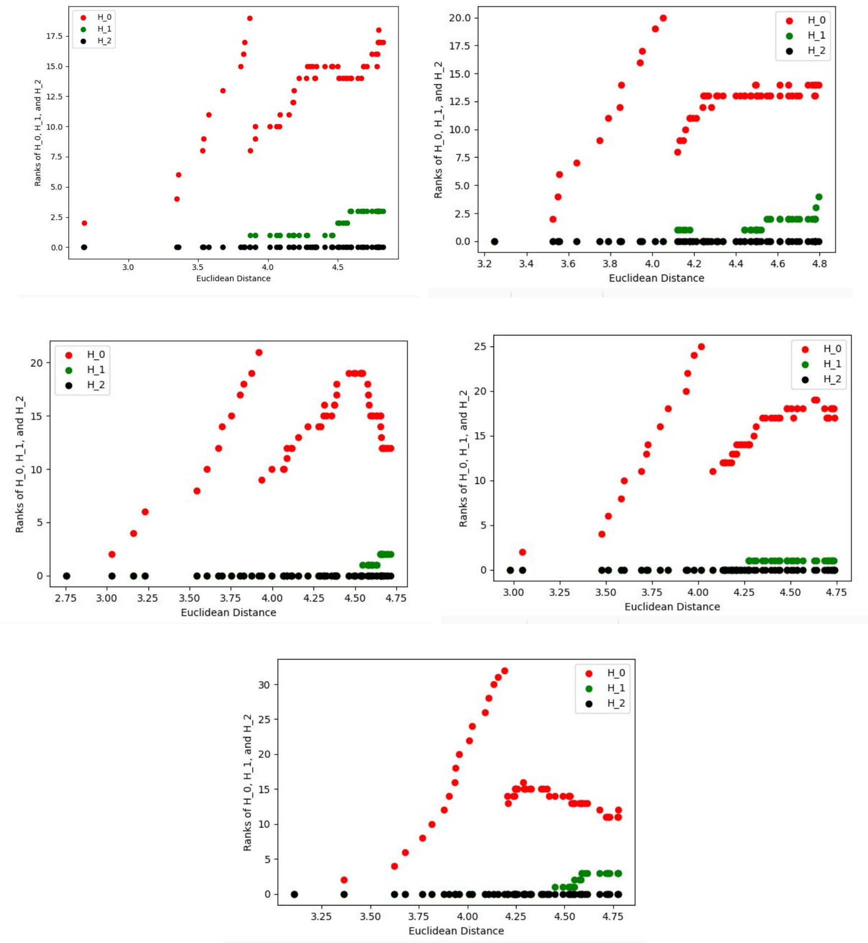

Figure 4: Persistent homologies on noisy datasets 1-5 (left-to-right).