STAT 37411 Homework 2

Due Feb 19, 2021 at 12pm

This homework covers material from weeks 2-4, primarily homology and persistent homology. If you get stuck on something, please ask for help via email or on Canvas before spending unnecessary time.

For programming problems you can use any programming language you would like and any TDA package you would like. This is so you can use whatever you are most comfortable with, or learn a TDA package in a language that is most likely to be useful to you. There is a list of some TDA packages on the Resources page of the course website. You can also use demonstration code from class if it is useful.

Please see the syllabus for the collaboration policy. Working with others is encouraged, and using the internet/search engines is allowed. Just state who you worked with and what references you found useful.

Jupyter notebooks can render quite a bit of math. You may find it convenient to just submit a Jupyter notebook. You can also do LaTeX/scripts, or whatever you like best as long as it is straightforward to understand.

Please upload any files for this assignment to Canvas. Alternatively, you can host your assignment on a personal webpage and submit a link.

When you write code, please provide comments/docstrings to make it readable.

Note on TDA Packages

In this homework, you'll want to use a TDA package which provides the following functionality (or implement it yourself)

- Construction of simplicial complexes
- Iteration over simplices of a simplicial complex
- Construction of filtered simplicial complexes (filtrations)
- Computation of persistent homology
- Iteration over persistence pairs (output of persistent homology)
- Visualization of persistence diagrams/barcodes

- Computation of bottleneck distance

Many packages support this. One example is Dionysus (`https://www.mrzv.org/software/dionysus2`). Note that in Dionysus, you can add simplices to a Filtration without giving a filtration value, in which case the simplex appears at parameter 0. You can treat a filtration in which all simplices are added in this manner as a simplicial complex, even though the type is technically a Filtration.

1 Chain Complexes as Quiver Representations

Recall that a chain complex is a diagram of vector spaces

$$0 \longleftarrow C_0 \xleftarrow{\partial_1} C_1 \xleftarrow{\partial_2} C_2 \longleftarrow \dots \tag{1}$$

where $\partial_k \circ \partial_{k+1} = 0$.

This is an example of a type-A quiver representation with a special condition on the linear maps.

1. What are the possible indecomposables of this quiver representation?
2. Characterize homology in dimension k using indecomposables.

2 Filtrations

Demonstrate the functions you write below on a simple example or two.

2.1 Lower-Star Filtrations

In the programming language and TDA package of your choice implement a *lower-star filtration*. The input of this filtration is a simplicial complex, and a function f on the vertices (0-simplices). The output is a filtration where the simplex $(i_0, \dots, i_k)$ appears at $\max_{i \in i_0,\dots,i_k} f(i)$ (i.e. this filtration is extended from a sub-levelset filtration on the 0-simplices).

If the TDA package you are using already implements this procedure, write your own. The important thing is to figure out how to iterate over simplices in a simplicial complex.

2.2 Rips Filtrations

In the programming language and TDA package of your choice implement a *Rips filtration*.

The input of this filtration should be a $n \times n$ matrix of pairwise distances, and a maximum simplex dimension k. The output should be a Rips complex on n vertices (0-simplices), and maximum simplex dimension k. You can modify your Rips complex implementation in the first homework.

Again, write your own version even if the TDA package you are using provides a procedure.

3 Persistent Homology

Sample 100 points uniformly at random from the unit circle in $\mathbb{R}^2$. Compute persistent homology of a Rips filtration using the Euclidean metric on this point cloud. Use $\mathbb{F}_2$ coefficients (this is the default in many TDA packages). You can use your Rips function above, or a function provided by a TDA package.

Visualize persistent homology in dimensions 0 and 1 using either persistence diagrams, persistence barcodes, or both. Write a sentence or two about what you see.

4 Bottleneck Distance

Generate 4 point clouds:

- Two point clouds of 100 points each sampled uniformly at random from the unit square
- Two point clouds of 100 points each sampled uniformly at random from the unit circle.

Compute persistent homology (with $\mathbb{F}_{\not\varepsilon}$ coefficients) in dimension 1 for the Rips filtration of each point cloud (note that you only need to go up to the 2-skeleton of the Rips complex for this). Compute a matrix of pairwise bottleneck distances between the persistence modules. What do you observe?

5 Data Analysis - Texture

In this problem, you will use persistent homology to build features for texture.

We'll use the SIPI rotated texture database (`http://sipi.usc.edu/database/database.php?volume=rotate`) which consists of grayscale images of textures rotated at different angles. This is a fairly clean and simple data set.

Note that the Python library PIL (or pillow) easily handles tiff files.

5.1 Construct a Simplicial Complex

Each image is 512×512 pixels. We'll build a simplicial complex on the square which has 512^2 0-simplices, which we can index as $(i, j) \in \{0, \ldots, 511\}^2$. We're triangulating a square domain, and will do so using the Freudenthal triangulation.

- The 1-simplices are all possible pairs of $((i, j), (i+1, j))$, $((i, j), (i, j+1))$, and $((i, j), (i+1, j+1))$.
- The 2-simplices are all possible triangles on the 1-skeleton. These simplices can take two forms: $((i, j), (i+1, j), (i+1, j+1))$ or $((i, j), (i, j+1), (i+1, j+1))$.

Write a procedure that constructs a the above simplicial complex.

Note that you can map the pair (i, j) to a single integer using row or column-major order. E.g $k = 512 * i + j$ for row-major, with the inverse procedure $i, j = divmod(k, 512)$ (assuming you are using a zero-indexed language).

5.2 Compute Persistent Homology

For each image in the SIPI rotation dataset, compute persistent homology using a lower-star filtration on the pixel values. Do this in homology dimensions 0 and 1. Use $\mathbb{F}_2$ coefficients (this is the default in many TDA packages).

You can either use your implementation of a lower-star filtration, or a built-in version in a TDA package. If you wrote your function in an interpreted language such as Python it will likely be much slower.

5.3 Compute Features

Compute the following feature for each persistence diagram

$$\frac{1}{256^2 N} \sum_{(b_i, d_i) \in dgm} (d_i - b_i)(b_i + d_i)$$

where N is the number of points in the diagram, the 256 normalizes pixel values to be between 0 and 1, and infinite bars are ignored (otherwise your dimension 0 feature will always be ∞). This gives us two features for each image (one for persistent homology in dimension 0, and one for persistent homology in dimension 1).

Create a scatter plot of these two features (so there is a point for each image in the data set), where each texture class gets a distinct marker type/color. Create a legend. What do you observe?

5.4 Discussion (Bonus)

This question is entirely optional, but is good to think about.

In this problem you developed features for texture based on persistent homology which (hopefully) demonstrate some ability to discriminate between different textures. No, we haven't gone through the work to build a full classifier (We don't really have much data anyways), but the plot you produce above should be reasonably convincing.

Compare this procedure to another procedure that might be used to discriminate between textures (e.g. CNNs, exploratory tools, etc.). What are the benefits and draw-backs of the persistent homology approach versus the other approach? You may wish to think about computational performance as well as things like interpretability, robustness, etc. This is fairly open-ended, so feel free to do write about what you think is most interesting or important.