Shape classification task via persistent homology

immediate

October 24, 2023

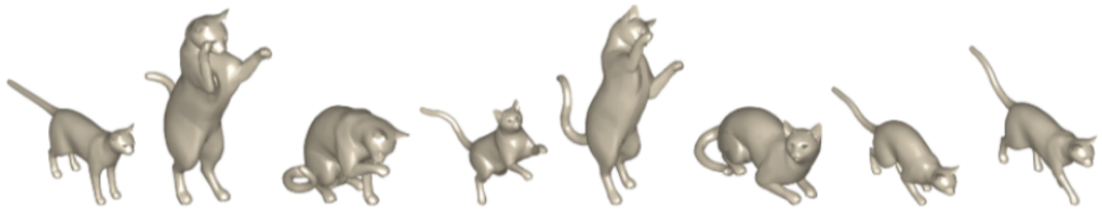
This project aims to give a brief hands-on experience at a real-world application of persistent homology. The dataset you will work with is the Non-Rigid World Tosca database of 3D shapes. This database has 148 different shapes represented as geometric simplicial complex from different classes of creatures (dogs, cats, horses, gorillas, etc.). There can be multiple shapes of the same creature with different poses among those 148 shapes.

The ultimate goal is to discriminate between the different creatures using persistent homology, regardless what pose they may be in.

Remark 0.1. You might wonder why we don't simply use the Gromov-Hausdorff distance for classifying these shapes. Unfortunately, computing the Gromov-Hausdorff distance is not practical; it is an NP-hard problem. On the other hand, computing persistent homology and the bottleneck distance can be accomplished in $O(n^3)$ time, where n represents the input size.

1 Dataset

The database (available on our KLMS) contains a total of 148 objects, including 9 cats, 11 dogs, 3 wolves, 17 horses, 15 lions, 21 gorillas, 1 shark, 24 female figures, and two different male figures, containing 15 and 20 poses, respectively. The database also contains 6 centaurs, and 6 seahorses. Each object contains approximately 3000 vertices. Three representations are available: MATLAB file (.mat) and ASCII text files containing the 1-based list of triangular faces (.tri), and a list of vertex XYZ coordinates (.vert). A .png thumbnail is available for each object. For each class (i.e. dogs, cats, horses, gorillas, etc) there are usually several different poses: The different poses of a cat in the database are illustrated below.



2 Preliminaries

In this section, we discuss the required mathematical concepts and computational software necessary to complete the project.

2.1 Mathematical concepts

You should know the following concepts that were covered (or will be covered) in classes:

1. Vietoris-Rips complex
2. Persistence diagrams
3. Bottleneck and Wasserstein distances between persistence diagrams

We also remark that stability of persistence diagrams (as to be established class) serves as a justification for the utilization of persistence diagrams in this project.

You should independently explore the following concepts:

1. Farthest point sampling (FPS) https://en.wikipedia.org/wiki/Farthest-first_traversal
2. Geodesic distance on a graph and Dijkstra's algorithm for computing it
    - https://en.wikipedia.org/wiki/Distance_(graph_theory)
    - https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm.
3. Multidimensional scaling (MDS); see the paragagrph *Details* in https://en.wikipedia.org/wiki/Multidimensional_scaling

2.2 Software

You may need to install the following (and more) on your computer:

1. Python;
2. Python library `Ripser` for computing Vietories-Rips persistent homology; https://ripser.scikit-tda.org/en/latest/;
3. Python libraries `GUDHI` and `POT` for computing the Wasserstein distance between persistence diagrams; https://gudhi.inria.fr/python/latest/wasserstein_distance_user.html

If you have not used Python before, you may find the online course 'Introduction to Programming (CS101)' at KAIST helpful. You can access it at the following URL: https://kooc.kaist.ac.kr/cs101/joinLectures/21778. The first lecture provides instructions on how to install Python. Feel free to seek assistance from the TA if needed.

3 Your task

You should download the database folder `nonrigid3d` from KLMS.

3.1 Preprocessing datasets

The initial step involves preprocessing the shapes to reduce the number of points from several thousands to a few hundred. This simplification allows `Ripser` to handle them computationally with ease. For each shape in the folder `nonrigid3d`:

1. Read the shape into Python.
2. Create a weighted graph representation and compute the geodesic distance matrix d_G on the graph (here, you exploit the given triangulation information).
3. Subsample 200 points via FPS using d_G.
4. Compute the Vietoris-Rips persistence diagrams in dimensions 0, 1, and 2 for d_G on the subsampled 200 points.
5. Create a struct called `shape` and attach fields `name`, `dm` (for the distance matrix).
6. Save the resulting struct to the folder named `processed_geodesic`. In this folder, you save the preprocessed version of `cat0.mat` as `pre_geodesic_cat0.mat` or `pre_geodesic_cat0.xxx`, depending on the file format you prefer. Use the same naming convention for other files.

After finishing all these, instead of Step 2 above, do the following:

2′. Compute the Euclidean distance matrix d_E on the point cloud (here, you do *not* exploit the triangulation information).

Then, follow the subsequent steps using d_E instead of d_G. In Step 6, you can save the preprocessed version of `cat0.mat` as `pre_euclidean_cat0.mat` and so on in the folder named `processed_euclidean`.

3.2 Distance matrix using a coarse geometric feature

Without any sophisticated tools, we can quickly perform a clustering using the diameter of a shape (defined as the maximal distance across all pairs of points). Try computing the matrix `dDiam` such that `dDiam`(i,j) equals $|\mathrm{diam}(S_i) - \mathrm{diam}(S_j)|$ for all shapes S_i and S_j. Save `dDiam` in a single file, in a common format like CSV or TXT.

Remark 3.1. Recall from your midterm that this distance, `dDiam`(i,j), is upper-bounded by twice the Gromov-Hausdorff distance between shapes S_i and S_j. Thus, although `dDiam`(i,j) is coarse, it is at least a *stable* dissimilarity measure in a sense.

3.3 Distance matrix based on d_E and Bottleneck distance

For any metric space (X, d_X), let $\mathrm{dgm}_p^{\mathrm{VR}}(X, d_X)$ be the p-th Vietoris-Rips persistence diagram of (X, d_X). Compute the distance matrix $D_{E,\infty}$ given by

$$D_{E,\infty}(i,j) := \max_{p\in\{0,1,2\}} d_B(\mathrm{dgm}_p^{\mathrm{VR}}(S_i, d_E), \mathrm{dgm}_p^{\mathrm{VR}}(S_j, d_E)),$$

and save in a single file.

3.4 Distance matrices based on d_G and Wasserstein distance

For $q \in [1, \infty]$, let $d_{W,q}$ be the q-Wasserstein distance between persistence diagrams. For each of $q = 1$ and $q = \infty$, compute the distance matrix $D_{G,q}$ given by

$$D_{G,q}(i,j) := \max_{p \in \{0,1,2\}} d_{W,q}(\mathrm{dgm}_p^{\mathrm{VR}}(S_i, d_G), \mathrm{dgm}_p^{\mathrm{VR}}(S_j, d_G)).$$

Save $D_{G,1}$ and $D_{G,\infty}$ in two separate files.

4 What you submit

You are asked to submit:

1. A report as a PDF file to Gradescope
2. A ZIP folder to KLMS

The report mentioned in Item 1 will be described below.

The ZIP folder of Item 2 should contain all files needed for producing the report. Namely, (1) all codes used for your tasks, (2) preprocessing results, and (3) computed distance matrices. Inside the ZIP folder, please include a document titled *Readme* that provides detailed descriptions of each file contained in the ZIP folder. In particular, the document should include a description of the structure of the datasets included in (2) and (3). The clarity of this document is of utmost importance for the evaluation of your submission.

The report should include 8 images and a brief discussion section, as described below.

1. A heat map image for the distance matrix `dDiam` from Section 3.2.
2. A heat map image for the distance matrix $D_{E,\infty}$ from Section 3.3.
3. (a) Heat maps, (b) Single-Linkage Hierarchical Clustering dendrograms, and (c) 2-dimensional MDS plots for the distance matrices $D_{G,q}$ with $q = 1$ and $q = \infty$. (note that you need 6 images in total from this item).

At the end of your pdf file, discuss the following.

- Is `dDiam` an effective dissimilarity measure for distinguishing between different shape classes?
- Can you envision a clustering scenario where shape diameters are valuable? You can try to imagine a realistic situation (that might be different from this project) where you might use `dDiam`, either alone or in combination with other dissimilarity measures, to improve clustering results or save time. You are encouraged to discuss this question with other people.
- Compare the heat maps of $D_{E,\infty}$ and $D_{G,\infty}$. Which one shows a more favorable clustering outcome? What is the criteria you used to determine which clustering outcome is superior (you may need to perform a search and then employ some relevant data analysis metrics to quantify the quality of your clustering results). Share your thoughts on why one gives a more favorable clustering outcome than the other.

- Among the clustering results presented in Sections 3.2, 3.3, and 3.4, which one produced the most favorable clustering outcome? What is your criteria? Consider what improvements could potentially enhance the best clustering method further. For such improvements, are there any associated costs or trade-offs to be aware of?

Please keep your answers concise and clear.

5 Criteria for Evaluation

1. **Completeness**: All required components are included.
2. **Integrity**: You do your work. It shouldn't be copied and pasted from your classmates work.
3. **Timeliness**: Submissions meet the deadlines.
4. **Clarity**: Explanations and presentations are clear.

You are welcome to consult various online resources as needed and seek assistance from ChatGPT. However, it is essential that you understand your code and how it works. Submitting a copy of a classmate's work is not allowed.