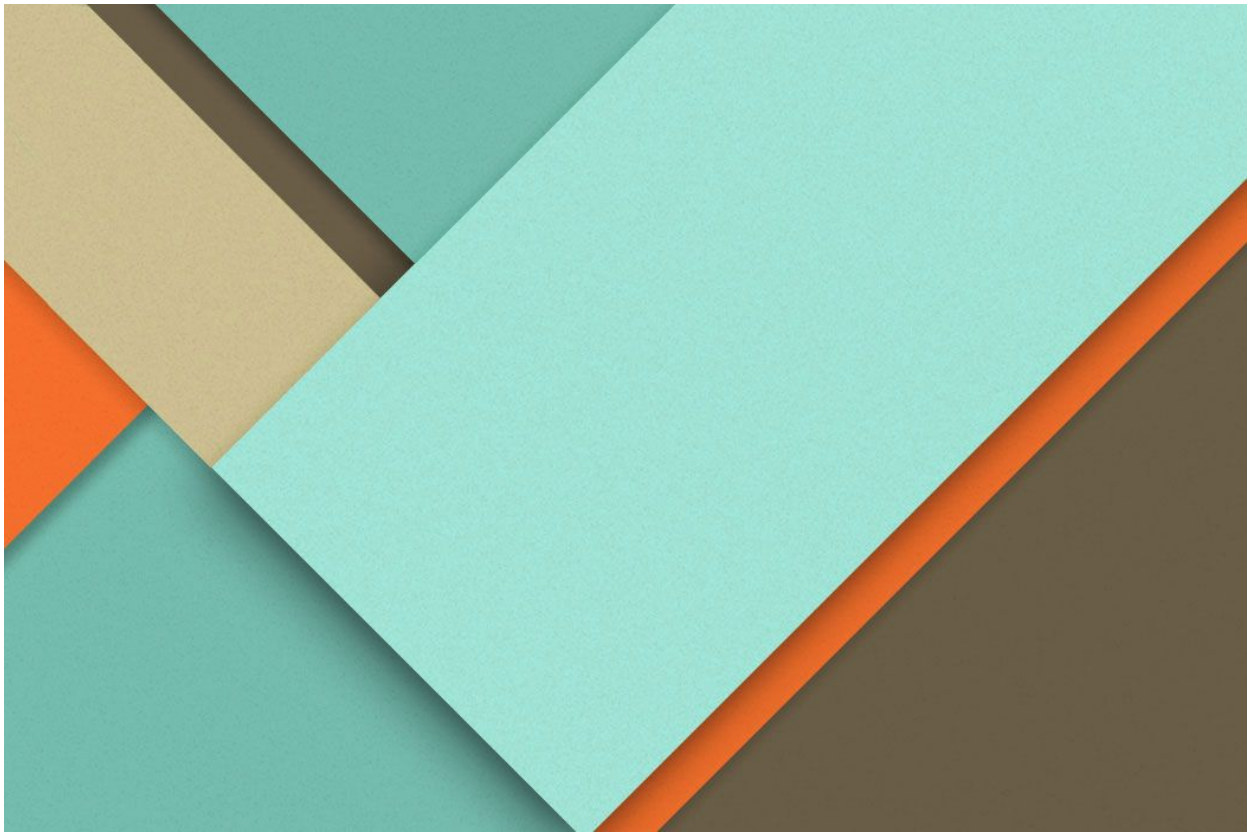

Applied topology project

23.06.2018

Berraj Omar

Overview

The dataset analyzed in this project, using persistent homology with aid of the javaplex software, is an image recognition dataset for developing unsupervised feature learning, deep learning, and self-taught learning algorithms. It is inspired by the CIFAR-10 dataset but with some modifications.

Specifications

The main data file to be loaded is 'test.mat', it contains 3 different sub data files. The one we're interested in is the 'X' table which contains random colored images' pixels as a matrix with 1 example per row. In each row, the pixels are laid out in column-major order, one channel at a time. That is, the first 96 * 96 values are the red channel, the next 96 * 96 are green, and the last 96 * 96 are blue, which are converted to RGB images for further use, but we're solely interested in the matrix 'X' for the sake of our analysis.

A submatrix A(8000, 80) is used for the analysis, giving that the original matrix X(8000, 27648) is huge, and the computation time takes too long.

Milestones

I. Source code:

```
clc; clear; close all;
import edu.stanford.math.plex4.*;

load test.mat
A = im2double(X(:, 1:80));
size(A)

max_dimension = 3;
num_landmark_points = 50;
nu = 1;
num_divisions = 1000;

% create a sequential maxmin landmark selector
landmark_selector = api.Plex4.createMaxMinSelector(A, num_landmark_points);
R = landmark_selector.getMaxDistanceFromPointsToLandmarks()
max_filtration_value = R / 3;

% create a lazy witness stream
stream = streams.impl.LazyWitnessStream(landmark_selector.getUnderlyingMetricSpace(), landmark_selector, max_dimension, max_filtration_value, nu,
num_divisions);
stream.finalizeStream()

% print out the size of the stream - will be quite large since the complex
% construction is very sensitive to the maximum filtration value
num_simplices = stream.getSize()

% get persistence algorithm over Z/2Z
persistence = api.Plex4.getModularSimplicialAlgorithm(max_dimension, 2);

% compute the intervals
intervals = persistence.computeIntervals(stream);

% create the barcode plots
options.filename = 'lazyRange';
options.max_filtration_value = max_filtration_value;
options.max_dimension = max_dimension - 1;
plot_barcodes(intervals, options);

%% DCT

figure;
scatter(A(:,1), A(:,50), '.')
axis square
```

II. Output plots:

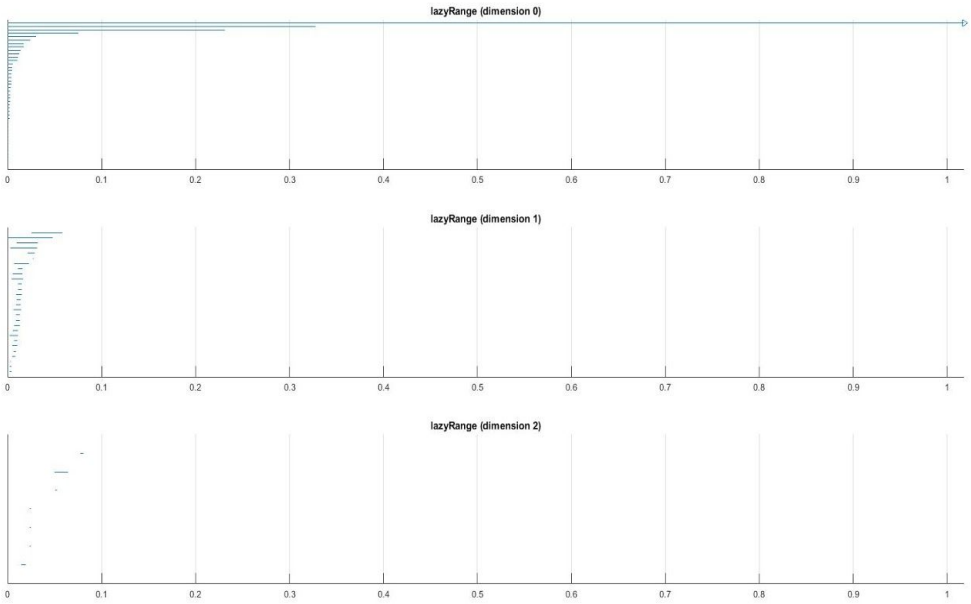

Figure 1: Betti intervals for the lazy witness complex built for A(8000, 80)

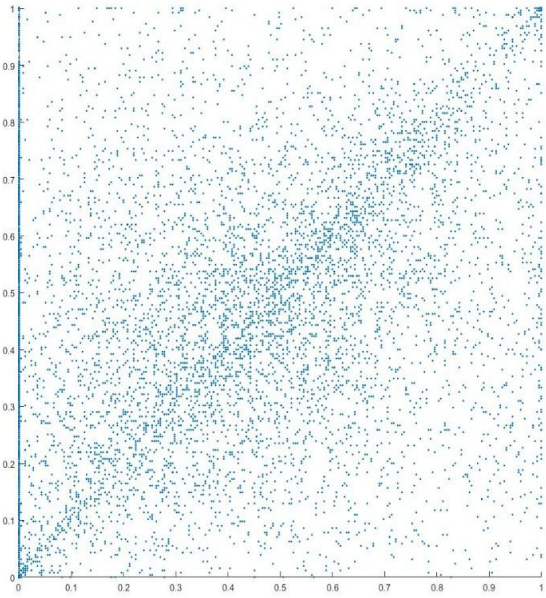

Figure 2: Projection of A(8000, 80)

III. Dataset plot:

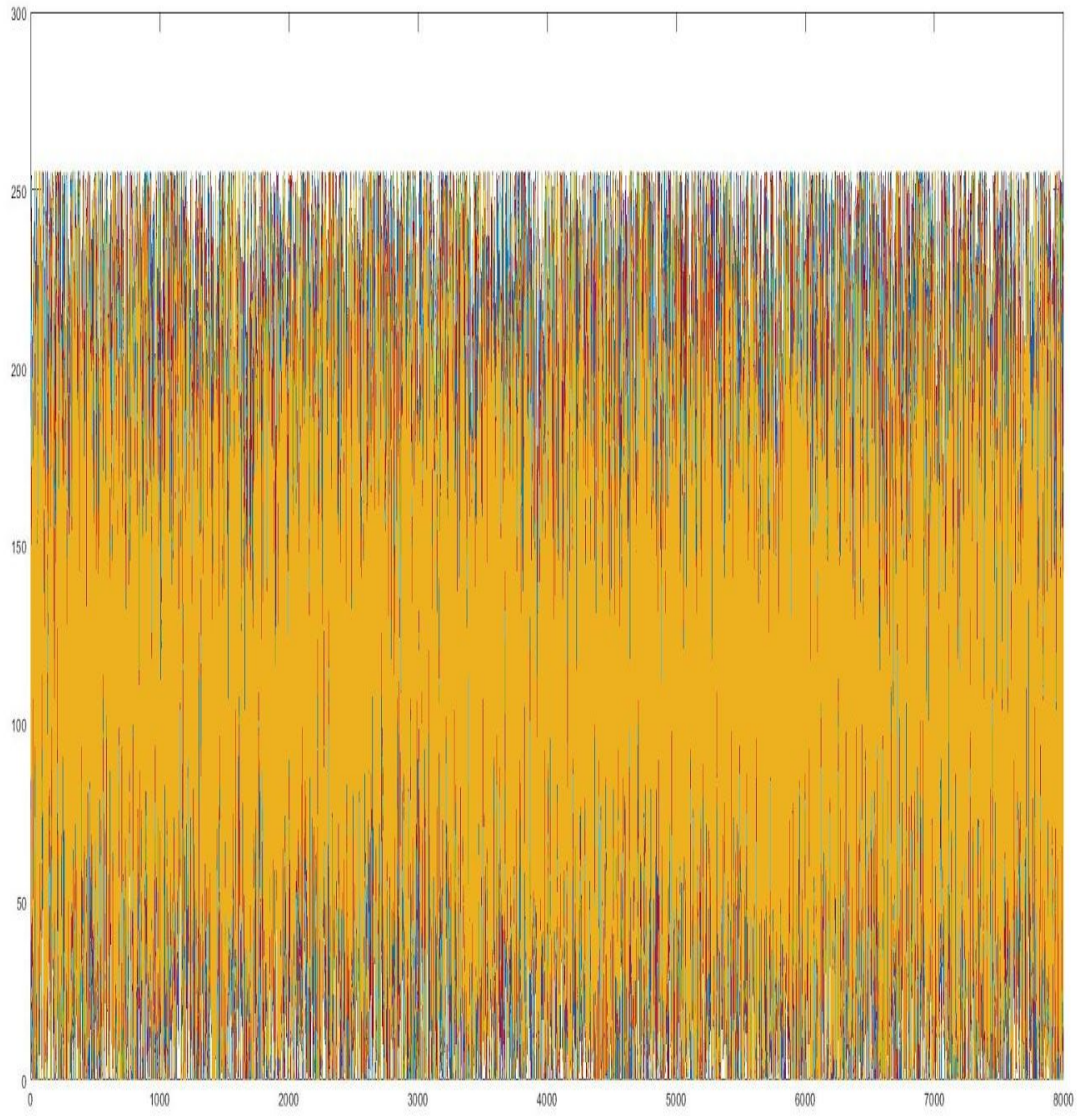

Figure 3: Plot of our dataset showing the different colors in the RGB model (0 -255)

IV. Sample image:

V. Description:

- We start by defining a few variables, first the 'max_dimension' which indicates the largest persistent homology dimension to be displayed. In our case the maximum dimension will be 3 (2-D space). We calculate the homology of our dataset in dimensions 0, 1 and 2.
- The 'num_landmark_points' variable (50 in our case), indicates how many data points to be chosen as a vertex per simplicial complex to avoid having every data point as a vertex (solution applicable to large datasets).
- The 'nu' variable typically chosen to be either 0, 1 or 2 determines how the lazy witness simplicial complexes are to be constructed.
- The 'num_divisions' variable is optional. If this input parameter is not specified, the the default value of 20 is used. The value determines, how many divisions are made to the maximum filtration value.
- Next, we select the landmark points using a greedy inductive selection process, called sequential maxmin. Landmark points picked using this method tend to cover the dataset and spread apart from each other.
- At this point, we also initiate a new variable 'R', which reflects how finely the landmarks cover the dataset. It is used as a guide for selecting the maximum filtration value for a lazy witness stream.
- Using the already defined variables, we can create a lazy witness stream which in-turn creates simplicial complexes from the landmark points passed to it. We finalize the stream readying it for the next step.
- The persistence object is used afterwards to compute the homology of our complexes given the dimension number and the coefficient, which is, in our case, Z/2Z, and the input can be any prime number.
- The next computation (intervals) is what gives us the betti intervals. They are computed with the persistence algorithm for representative intervals.
- The Matlab function 'plot_function' lets us display the intervals as Betti barcodes, after choosing a filename, the maximum filtration value and the maximum dimension.
- The final figure shows a subset of the points in our dataset scattered on a plot within a certain range.

VI. Meaning of the output:

- From the dataset plot, we can clearly see that the pixels of the images, share a common color interval in the RGB values (orange being the dominant color).
- The betti_k number of the stream (simplicial complex) at filtration value t is the number of intervals in the dimension k plot that intersects a vertical line through t.
- Given the size of our dataset, and since we're including a large number of vertices even in the submatrix A, we've chosen to use the witness lazy stream to address this problem.
- The lazy witness stream, in building streams (simplicial complexes), selects a subset of the vertices called landmark points, as the only vertices.
- From figure 1 we can deduct the following:
    - Take into account that our stream contains more than 100000 simplices using the lazy witness stream. At value 0 multiple connected components appear in dimension 0 and a 1-D hole appears in dimension 1.
    - As the value of t increase, the components which appeared at value 0 each join the component above them in dimension 0.
    - At value 0.18, all the previous component have joined a singled connected component in dimension 0.
    - It is clear that the computed intervals do not persist necessarily until t = inf.
    - In relation to persistent homology, we have a single long interval which we can consider as a real topological feature whereas the rest of the small intervals can be considered as noise.
    - For the long range mentioned, the Betti numbers betti_0 = 1, betti_1 = 0 and betti_1 = 0 are obtained, meaning the rank of the 0th homology group of A is 1, and the rest being 0.
    - This is evidence that the core of our data is approximated by a single connected space.
- The connected space we receive is projected in Figure 2:
    - It shows a projection of A(8000, 80).
    - Our dataset satisfies the criteria of point cloud data, that is data modeled by a finite metric space.
    - We can see from Figure 2, that our data is essentially a finite set of points equipped with a notion of distance.
    - All the data points in our initial set help serve as witnesses for the inclusion of higher dimensional simplices.
    - The disadvantage is that outlier points may be chosen, but given the dense core of our dataset, we can argue that outliers won't cause an issue.

VII. Summary:

- From the dataset plot, which is a plot of pixels (0 -255) of the RGB model, we can see that most pictures share a common pixel interval (80 - 150), the color modeled seems to be orange or a close relative. Those approximations are also seen in our analysis output plot (Figure 2), where the bulk of the points are concentrated around the middle forming a line going through the origin. Meaning, we can project our results onto the whole set of images giving that thanks to persistent homology, we can say that this single long interval is the real topological approximation of our dataset.

VIII. References:

- https://github.com/appliedtopology/javaplex
- https://cs.stanford.edu/~acoates/stl10/
- Adam Coates, Honglak Lee, Andrew Y. Ng An Analysis of Single Layer Networks in Unsupervised Feature Learning AISTATS, 2011.
- https://www.sciencedirect.com/science/article/pii/S0047259X1000117X