Project 2 - CS 6170

Nithin Chalapathi

Spring 2019

Data

I went with 8 images from 10 separate classes from the MPEG dataset. I intentionally picked a few classes that would easily be mixed up (i.e. two classes that both have a large dim1 loop).

- Apple
- Bird
- Bone
- Children
- Face
- Fork
- Horseshoe
- Pencil
- Spoon
- Watch

I expect that the apple and face classes will get mixed up due to a similar loop structure. I also believe that the fork, pencil, spoon, watch, and bone classes will also be somewhat mixed together due to the elongated nature of the items.

1 Barcodes and Projections

1.1 Preprocessing

I used the same procedure I used in project 1 to extract the boundary points of the image. For each image, I first converted it into a PNG (PNGs under "/part1/data/original_png/" and original GIF files under "/part1/data/originals/").

I treated the PNG matrices as a binary image, with grey pixels as white. I sampled a maximum of 1000 points from each image. No interior points where included in the extracted point cloud. Each point cloud is then saved under "/part1/data/point_clouds/" I used the python subprocess module to call ripser on each point cloud and saved the outputs in "/part1/data/ripser_outputs/". Before saving the persistent diagrams, I cleaned the output by removing any brackets, commas, and parentheses, in addition to Ripser's summary of the persistent diagram. Each diagram was also separated by dimension when being saved.

Note: All python code will be using python 3.6.

1.2 t-SNE and MDS

For each dimmenion, I computed the Wasserstein distance and the bottleneck distance by calling hera from my python script in a similar fashion to calling ripser. I saved the outputs as .npy files. For the raw distance matrices, there is a python script (/part1/data/print_distance_matrix.py) that will print the full arrays. For the graphs below, each class is color coded:

1. apple = blue
2. bird = green
3. bone = red
4. children = cyan
5. face = magenta
6. fork = yellow
7. horseshoe = black
8. pencil = grey
9. spoon = pink
10. watch = orange

Both figures will be saved as .PNG files under /part1/data/.

In the case of the Wasserstein distance graphs, there is very little clustering with dimension 0 and the MDS projection. Most of the points are bunched very close together and even when zoomed in, there doesn't appear to be a consistent ordering. However, the t-SNE project has some clustering that is defined. For example, children (cyan), spoon (pink), face (magenta), and bone (red) all have relatively well behaved clusters. There are a few classes that are don't cluster well including the horseshoe (black) and pencil (grey) classes.

In dimension 1, MDS produces better results than in dimension 0. The bone (red), watch (orange), and apple (blue) all have fairly well defined clusters.

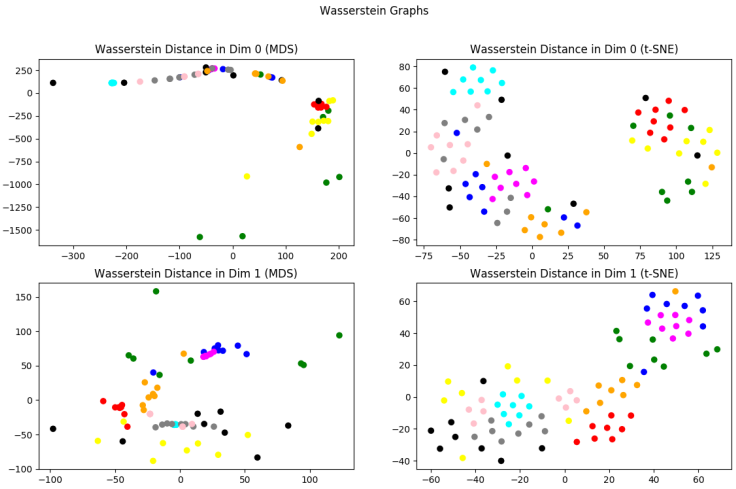

Figure 1: Wasserstein distance

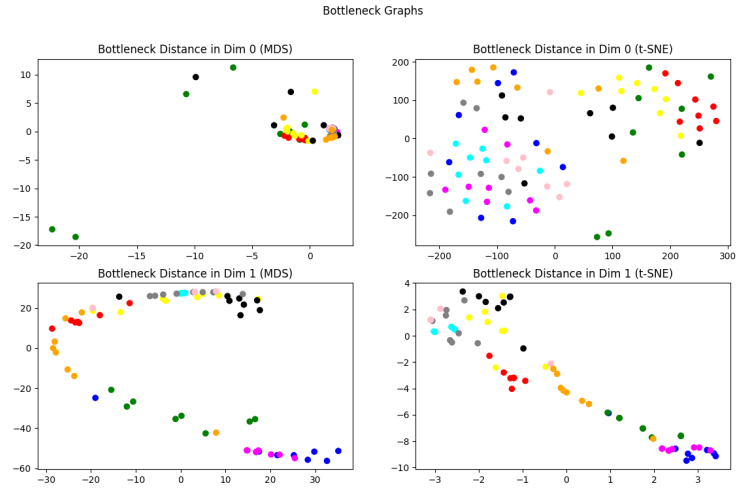

Figure 2: Bottleneck distance

However, there is still a lot of interweaving between clusters. t-SNE paints a similar picture but with less overlapping clusters when compared to the MDS projection.

When looking at the bottleneck distance graphs, there still isn't a well defined clustering structure. In dimension 0, the MDS projection clusters all of

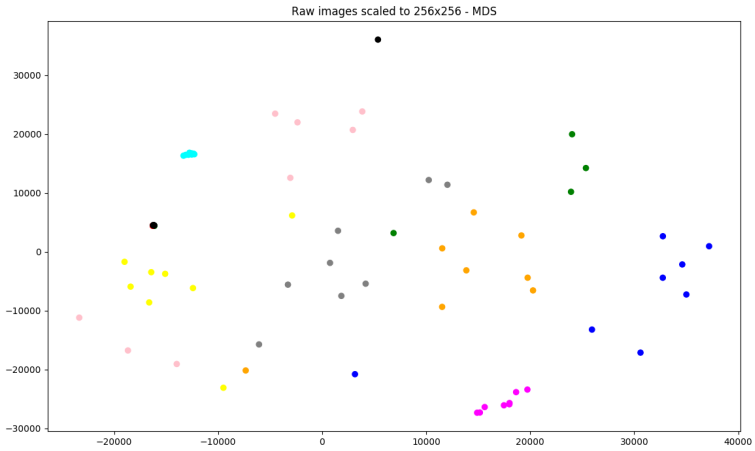

Figure 3: Raw image projection - MDS

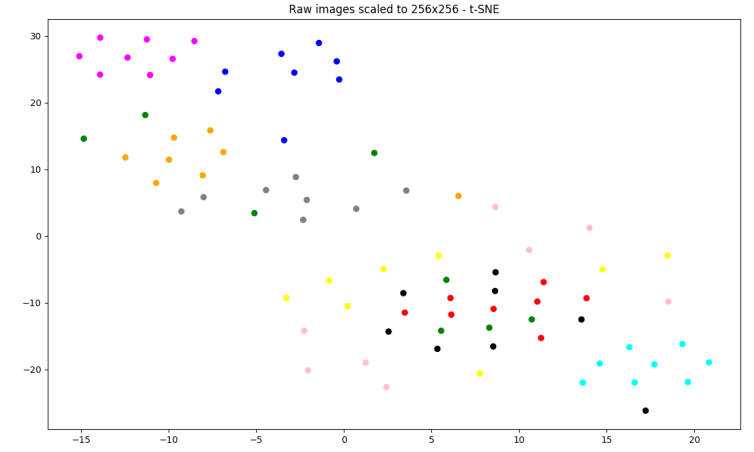

Figure 4: Raw image projection - t-SNE

the data points in the same area. t-SNE has a little bit of structure with bone (red) clustering relatively well. The rest of the points are grouped together. In dimension 1, the MDS projection seems to have a structure by creating clusters that follow a parabolic curve rotated 90 degrees clockwise. The t-SNE project also reflects this structure by having a linear trend correlated with class.

When looking at these trends at a high level, they reveal some insight into the data that I picked. I knew that going in I had classes that were topologically similar and geometrically, had similar shapes (like a bone and pencil). The graphs reinforce the idea that the images have similar structure and that training the SVM with topological signatures in part 2 might not yield the best results.

Additionally, the raw image projections are also available in figures 3 and 4. When projected with t-SNE, most of the classes were well clustered with the exception of the fork (yellow) and spoon (pink). In the MDS projection, a very similar situation occurs with the difference being only spoon (pink) does not have a well defined cluster.

Comparing the raw image projection with the distance projections, it seems as though the raw images have a better clustering habit. The raw projections were better able to capture more difficult to cluster classes such as horseshoe (black) and pencil (grey). However, the clustering of the bird (green) class suffered heavily when compared to the bottleneck and Wasserstein projections. This is especially apparent in the t-SNE projection of the Wasserstein distance in dimension 1. The bird class makes a much better cluster here than it does in the raw projections.

2 SVM Classification

2.1 Tools and methodology

I used the Sklearn SVC as the SVM. The kernels I used came from https://github.com/MathieuCarriere/sklearn_tda. The python script inside of /part2/src/ uses those libraries and runs a series of tests and outputs the best and worst validation accuracies along with their respective training accuracies. When doing cross validation, I used a 90 / 10 split of the data where 90% of the data was used for training and the other 10% was used for validation. This was then used to find the optimal C value (Sklearn parameter for strength of penalty for wrongly classified points).

For persistent images, I tested each of the following resolutions: 2x2, 4x4, 8x8, 16x16, 32x32, 64x64, 128x128, and 256x256. I tested dimension 0, dimension 1, a linear combination of the two (both weighted equally), and a concatenated feature vector with both images. For the persistent scale kernel, I tested dimension 0, dimension 1, and a linear combination of the two distances (weighted equally).

2.2 Results

The results and output can be found under /part2/results.pdf. The most interesting part of all the results was that the raw image classification was actually the best classifier, coming in at 90% validation accuracy. Given the clustering structures from part 1, this is unsurprising since the topological features I used in the SVM were lossy data summaries.

I also found it interesting how an 8x8 persistent image was enough to get a 60% validation accuracy. I would've expected it to be significantly lower given how small the resolution is. On the flip side, the 256x256 resolution did poorer than I thought with a validation accuracy of only 60%. This could be attributed to the topological features extracted and how they didn't provide as much clustering information as the raw images did.

3 The Code

3.1 MPEG Processing

The mpeg-processing.py under /part1/src/ takes a variety of different arguments. At least one must be specified and the order of arguments doesn't matter:

1. -gen-png: Creates pngs from the gif files in the original dataset and saves them.
2. -gen-point-cloud: Runs the boundary extraction on png images and saves the point clouds
3. -run-ripser: Runs ripser on saved point clouds and after cleaning up the output, saves it to a text file.
4. -run-hera-and-plot: Runs hera on the persistent diagrams and plots them. The computation is only ran if a saved version of the matrix can't be found.
5. -vis-mpeg: Plots the original png images using MDS and t-SNE.
6. -all: Runs all routines.

3.2 SVM and distance matrix printing

Both the SVM code under /part2/src/ and the distance printing script under /part1/data/ both do not take arguments. Both also assume that the script from part 1 has been run and all the necessary ripser output and hera output exists.

4 Raw SVM output

Persistent Images SVM in dim0: Resolution: 2x2
Optimal C: 1.7
Best testing accuracy: 0.4 with training accuracy of: 0.3
Worst testing accuracy: 0.3 with training accuracy of: 0.4

Persistent Images SVM in dim1: Resolution: 2x2
Optimal C: 1.7
Best testing accuracy: 0.4 with training accuracy of: 0.3
Worst testing accuracy: 0.3 with training accuracy of: 0.4

Persistent Images SVM in dim0 and dim1 by concat: Resolution: 2x2
Optimal C: 1.7
Best testing accuracy: 0.2 with training accuracy of: 0.15
Worst testing accuracy: 0.15 with training accuracy of: 0.2

Persistent Images SVM in dim0 and dim1 by sum: Resolution: 2x2
Optimal C: 1.7
Best testing accuracy: 0.4 with training accuracy of: 0.3
Worst testing accuracy: 0.3 with training accuracy of: 0.4

Persistent Images SVM in dim0: Resolution: 4x4
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.45714285714285713
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim1: Resolution: 4x4
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.45714285714285713
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim0 and dim1 by concat: Resolution: 4x4
Optimal C: 1.7
Best testing accuracy: 0.3 with training accuracy of: 0.22857142857142856
Worst testing accuracy: 0.15 with training accuracy of: 0.2571428571428571

Persistent Images SVM in dim0 and dim1 by sum: Resolution: 4x4
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.45714285714285713
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim0: Resolution: 8x8
Optimal C: 1.7

Best testing accuracy: 0.6 with training accuracy of: 0.4
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim1: Resolution: 8x8
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.4
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim0 and dim1 by concat: Resolution: 8x8
Optimal C: 1.7
Best testing accuracy: 0.3 with training accuracy of: 0.2
Worst testing accuracy: 0.15 with training accuracy of: 0.2642857142857143

Persistent Images SVM in dim0 and dim1 by sum: Resolution: 8x8
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.4
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim0: Resolution: 16x16
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.4
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim1: Resolution: 16x16
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.4
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim0 and dim1 by concat: Resolution: 16x16
Optimal C: 1.7
Best testing accuracy: 0.3 with training accuracy of: 0.2
Worst testing accuracy: 0.15 with training accuracy of: 0.2571428571428571

Persistent Images SVM in dim0 and dim1 by sum: Resolution: 16x16
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.4
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim0: Resolution: 32x32
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.4142857142857143
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim1: Resolution: 32x32
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.4142857142857143
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim0 and dim1 by concat: Resolution: 32x32
Optimal C: 1.7
Best testing accuracy: 0.3 with training accuracy of: 0.20714285714285716
Worst testing accuracy: 0.15 with training accuracy of: 0.2571428571428571

Persistent Images SVM in dim0 and dim1 by sum: Resolution: 32x32
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.4142857142857143
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim0: Resolution: 64x64
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.4142857142857143
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim1: Resolution: 64x64
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.4142857142857143
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim0 and dim1 by concat: Resolution: 64x64
Optimal C: 1.7
Best testing accuracy: 0.3 with training accuracy of: 0.20714285714285716
Worst testing accuracy: 0.15 with training accuracy of: 0.2642857142857143

Persistent Images SVM in dim0 and dim1 by sum: Resolution: 64x64
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.4142857142857143
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim0: Resolution: 128x128
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.4142857142857143
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim1: Resolution: 128x128
Optimal C: 1.7

Best testing accuracy: 0.6 with training accuracy of: 0.4142857142857143
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim0 and dim1 by concat: Resolution: 128x128
Optimal C: 1.7
Best testing accuracy: 0.3 with training accuracy of: 0.20714285714285716
Worst testing accuracy: 0.15 with training accuracy of: 0.2571428571428571

Persistent Images SVM in dim0 and dim1 by sum: Resolution: 128x128
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.4142857142857143
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim0: Resolution: 256x256
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.4142857142857143
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim1: Resolution: 256x256
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.4142857142857143
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Persistent Images SVM in dim0 and dim1 by concat: Resolution: 256x256
Optimal C: 1.7
Best testing accuracy: 0.3 with training accuracy of: 0.20714285714285716
Worst testing accuracy: 0.15 with training accuracy of: 0.2571428571428571

Persistent Images SVM in dim0 and dim1 by sum: Resolution: 256x256
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.4142857142857143
Worst testing accuracy: 0.3 with training accuracy of: 0.5142857142857142

Raw image classification - RBF kernel with gamma='scale'
Optimal C: 1.7
Best testing accuracy: 0.9 with training accuracy of: 0.8571428571428571
Worst testing accuracy: 0.6 with training accuracy of: 0.8571428571428571

PSS kernel SVM in dim0
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.18571428571428572
Worst testing accuracy: 0.4 with training accuracy of: 0.17142857142857143

PSS kernel SVM in dim1
Optimal C: 1.7
Best testing accuracy: 0.7 with training accuracy of: 0.42857142857142855
Worst testing accuracy: 0.4 with training accuracy of: 0.4857142857142857

PSS kernel SVM with dim0 and dim1
Optimal C: 1.7
Best testing accuracy: 0.6 with training accuracy of: 0.18571428571428572
Worst testing accuracy: 0.4 with training accuracy of: 0.17142857142857143