Visualization
Prof. Bernhard Schmitzer, Uni Göttingen, summer term 2024

Problem sheet 6

- *Submission by **2024-06-04** 18:00 via StudIP **as a single PDF/ZIP** (one day extension due to bank holiday on Monday). Please combine all results into one PDF or archive. If you work in another format (markdown, jupyter notebooks), add a PDF converted version to your submission.*
- *Use Python 3 for the programming tasks as shown in the lecture. If you cannot install Python on your system, the GWDG jupyter server at https://jupyter-cloud.gwdg.de/ might help. Your submission should contain the final images as well as the code that was used to generate them.*
- *Work in groups of up to three. Clearly indicate names and enrollment numbers of all group members at the beginning of the submission.*

Exercise 6.1: dimensionality reduction on histograms.
The file `hist.npz` contains a small histogram dataset of 199 histograms with labels. The key `data` contains the normalized histograms, the key `labels` contains the binary labels (0 or 1), the key `binspec` contains the boundaries of the histogram bins. Analyze this dataset with PCA, as discussed in the lecture. Use both the standard Euclidean metric and the inverse cumulative distribution transform for the analysis an compare the results. Concretely, proceed as follows:

1. Import the histograms and visualize them with their labels.
2. Apply PCA to the histograms, show the spectrum of eigenvalues, and plot their embedding obtained by projection onto the first two eigenvectors with label information. Do the two classes appear to be different?
3. Apply the inverse cumulative distance transform as in the lecture and repeat the same steps as above. How does the spectrum change?
4. For both versions of the data, visualize the change of histograms corresponding to moving along the first PCA direction by one standard deviation. Briefly describe the induced change in the histograms.

Exercise 6.2: nonlinear manifold learning with UMAP.
The directory `imgs/` contains a large number of small PNG images of a sandal.[1] Each filename is of the form '`sandal_e{x}_a{y}.png`' where `x` and `y` are integers that specify the *elevation* and *azimuth* angles that were used in the rendering of the mesh.[2] The size of the images is 100 by 100 pixels. The images were generated by varying two underlying parameters. This means, we expect that the images roughly form a two-dimensional sub-manifold of $\mathbb{R}^{100\times100}$. In the following, we will test how well UMAP can extract this.

[1] source for the original 3d data: https://raw.githubusercontent.com/plotly/datasets/master/ply/sandal-ply.csv
[2] see https://matplotlib.org/stable/api/toolkits/mplot3d/axes3d.html for documentation on view angles

1. Import all images into Python as 2d numpy arrays and extract the corresponding values for *elevation* and *azimuth* from the filenames. What are the value ranges for *elevation* and *azimuth* in the data?
   Hint: The functions `glob.glob`, `imageio.imread`, and the module `re` could be useful for this.

2. Analogous to the lecture, use UMAP to obtain a two-dimensional embedding of the images. For this, treat all images as flattened vectors of length 10,000. Plot the embedding coordinates in a scatter plot and additionally encode the *elevation* and *azimuth* parameters of the images as additional visual attributes. How well does UMAP recover the underlying manifold?

3. UMAP uses stochastic optimization. Therefore, each run will yield slightly different results. Generate the embedding plot 5 (or more) times and comment on the observed differences.

4. (This last bullet point is purely voluntary and not part of the problem sheet grading.) UMAP uses the standard Euclidean distance as distance between images. This is not very stable. We can stabilize the results a little by applying a blur convolution first. Let

   $$k = \begin{pmatrix} 1/16 & 1/8 & 1/16 \\ 1/8 & 1/4 & 1/8 \\ 1/16 & 1/8 & 1/16 \end{pmatrix}$$

   be a simple blur kernel. This can be applied to the original images via

   ```
   img_blurred=scipy.signal.convolve2d(img,kernel,mode="valid")
   ```

   to generate slightly blurred images of size 98×98. (Here `img` is the original 100×100 image.) Apply this to all images and then re-compute and re-plot the UMAP embeddings as above. Did the results improve?