INF264 - Homework 6

Natacha Galmiche

Note that a template is provided to help you with technical details.

1 Hierarchical clustering

In this exercise, we will use hierarchical clustering in order to visualize clusters hierarchy produced on a small dataset representing the 3D coordinates of 30 international cities. We will see how different linkages can affect clusters hierarchy. Visualization will be achieved via dendrograms, which are trees showing the order and distances of clusters merges during hierarchical clustering.

1. Load the dataset contained in the file `cities_coordinates.txt`. Store the features (the xyz coordinates of the cities) in a matrix `X` and the names of the cities in a vector y.
2. Perform hierarchical clustering on `X`, for different types of linkage: 'ward', 'average', 'complete' and 'single'. You can use `sklearn.cluster.AgglomerativeClustering`.
3. For each linkage, plot the dendograms. You can use the `scipy.cluster.hierarchy.dendrogram` and see a example here: `https://scikit-learn.org/stable/auto_examples/cluster/plot_agglomerative_dendrogram.html`.
4. For each linkage type, display a 3D scatterplot of the cities' positions, where each city has its 3D point colored like its cluster in the dendrogram. From the observation of your 3D scatterplots, which linkage type seems to be most/least realistic?

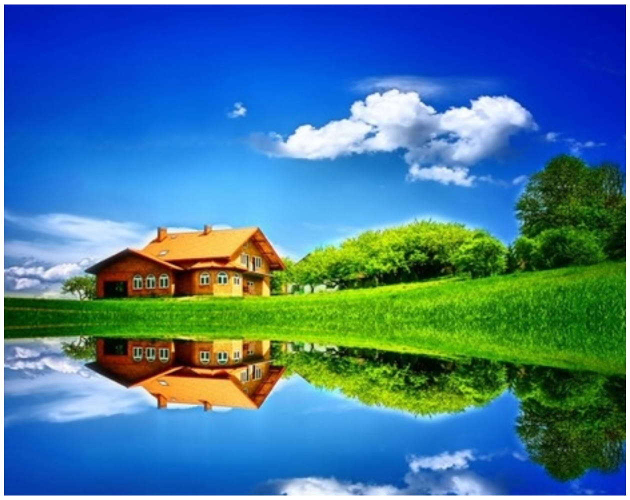
2 An application of clustering to image segmentation

In this exercise, we will apply a clustering algorithm to solve the problem of image segmentation. Image segmentation aims at partitioning an image into sets of similar pixels, in order to obtain a meaningful simplified representation of the image.

We consider an image represented by a three-dimentional matrix in $\mathbb{R}^{m\times n\times d}$, where m is the number of rows of pixels (so the "height" of the image), n is the number of columns of pixels (so the "width" of the image) and d is the number of channels (1 for grayscale images, 3 for RGB images). This means that an image has $m \times n$ pixels, where each pixel is a vector in $\mathbb{R}^d$.

A natural way to measure the similarity between two pixels p_1 and p_2 is simply to compute their euclidean distance in $\mathbb{R}^d$, that is $\|p_1 - p_2\|_{L_2}^2$. We thus can reduce the image segmentation problem to K-means: all we have to do is reshaping the image matrix into a two-dimentional matrix in $\mathbb{R}^{(m\cdot n)\times d}$, meaning the image is considered as a dataset and each pixel in the image is seen as an observation in $\mathbb{R}^d$.

We will consider the following image represented by a matrix in $\mathbb{R}^{184\times 233\times 3}$:



1. Load the dataset contained in the file `scenery_184_233.txt` in a matrix `X`. Note that the `Scenery.jpg` image was vectorized into a $\mathbb{R}^{(184\cdot 233)\times 3}$ matrix, then converted into a `.txt` file where each line gives the RGB values of a pixel.
2. Make a copy `X_vis` of `X`, (this is only for visualization), reshape `X_vis` into a $\mathbb{R}^{184\times 233\times 3}$ matrix, then convert it to an array of integers. You can use for example `uint8` format using `numpy.uint8`.
3. Plot `X_vis` using `matplotlib.pyplot.imshow`.
4. Perform clustering on `X`, for different values of k. You can use `sklearn.cluster.KMeans`.
5. Plot the elbow graph. Which value of k would you choose?
6. Visualizing the "angle" of the elbow is not always easy, but it can be computed, using for instance `KneeLocator()` from the `kneed` library. In our specific case, the keyword arguments should be the following: `curve="convex", direction="decreasing"`.
7. Perform once more clustering on `X` with the value of k that you selected in the previous question. Also return a 1-dimensional predictions array containing the clusters' index for every pixel. You can do all that at the same time using the `fit_predict` method.
8. Each computed cluster can be seen as a "mask", where each pixel belonging to that cluster is represented by the centroid of that cluster. Separate the predictions into k sub-masks, where every pixel in the j-th sub-mask has value 1 if the same pixel is equal to j in the predictions mask, and 0 otherwise. Visualize each of these submasks.
9. Segment the `Scenery.jpg` image according to the computed sub-masks and display the resulting image.
10. Are you satisfied by the number of masks selected by the elbow method? Plot the segmented representation of `Scenery.jpg` for different values of k. Are there k values that seem more satisfying?