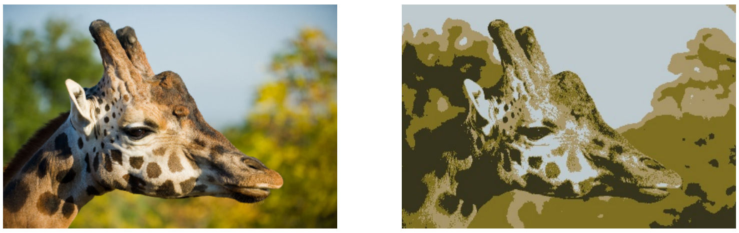
Practical Training - Task Sheet 1
Implementation of Clustering Algorithms

This is the first out of two practical training tasks that you will have to fulfill to qualify for the final exam. Work is to be done in groups of two.

1 Task Description

The task deals with the implementation of the two clustering algorithms `k-means` and `DBSCAN` (see [2] and [1]). As a working example, the segmentation of color images using clustering algorithms will be used. The objects that are to be clustered are the pixels of a given image. Each pixel is represented by its vector in the RGB color space (3-dimensional vector).
For the task, you are given a template containing an incomplete program with which the pixels of input images are supposed to be clustered. The output of the progam will be the provided image with its pixels' colors replaced by the respective cluster colors. An example can be seen in Fig. 1:



(a) Non-segmented image (b) Segmented image

Figure 1: Exemplary segmentation

1. Complete the program by implementing the functions `fit()` and `__init__()` for each class in the file `algorithms.py`! For the algorithms, the following parameters should be available and arbitrarily selectable:
   - `k-means`: `k` $\in \mathbb{N}$
   - `DBSCAN`: `eps` $\in \mathbb{R}$, `min_samples` $\in \mathbb{N}$, `metric` $\in \{$`'euclidean'`, `'manhattan'`, `'chebyshev'`$\}$
2. Implement distance functions based on the Euclidean, Manhattan and Chebyshev norm! Make them usable through the `metric` parameter of your DBSCAN implementation!

The template files can be found on the Moodle page.

2 Submission

1. The submission must be done before deadline given on the Moodle page.
2. The submission is to be done in Moodle. The submission link deactivates after the deadline. Submission is only possible if you are registered in a group.
3. Submit all files that are required to run and test your implementation. If necessary, include documentation documents.

3 Requirements

In order to pass, you and your submitted program must fulfill *all* of the following criteria:

1. All points listed under 'Submission' have to be fulfilled.
2. Using the source code and potential documentation documents, the program must be compilable without errors and run without major crashes.
3. The implemented algorithms have to work correctly (according to their underlying theory) and in a reasonable amount of time. They must pass the unit tests prepared by the teaching staff.
4. The implemented algorithms have to be in a file named `algorithms.py` and must be compatible with `image_clustering.ipynb` so that this file can be run without changes.
5. The clustering algorithms have to be implemented by you, personally. Detected plagiarism will result in immediate failure of the task.
6. The clustering algorithms should be implemented in a general way, so that they work with any given feature vectors in any dimensionality and not just with specified example data.
7. If requested, you must be able to explain *every* part of the source code, even segments that were implemented by your partner.
8. Comments and identifiers must be in the English language.

4 FAQ

Programming is not part of my curriculum. How can I do this task? This class requires rudimentary programming skills. However, programming itself is not our focus, thus code quality or efficiency are neglectable to a certain degree. Instead, it is important that you understand the clustering concepts and can apply them.

Which programming languages are allowed? The mandatory usage of our template dictates the programming language you may use: Python (version ≥ 3.0).

Is the loading and saving of the image part of the task? This functionality is built into the template already, thus you do not have to create it yourself.

How fast does the program have to be? Depending on your image, it is normal that the clustering (especially DBSCAN) can take a long time to terminate. As an estimation, clustering of a small image, e. g. 100x100 pixels, should not take longer than five minutes. (Although it should usually be faster than that.) If your program runs significantly longer than this, you should revise your code. If it never terminates, the task is not fulfilled.

The task demands that our clustering should work universally. Does that mean that the program has to accept input other than images? The task mentions only that the *algorithms* must be universal. The given template is still processing images. Only the clustering functions, that you are writing, should be universal. Think of it as writing your own clustering library.
Do not forget, that different data can have more or less than three dimensions. You can assume that all points in one data set have the same dimensionality, though. In practice, this means that you should assume an arbitrary `numpy` array as an input.

What about the alpha values of the images? Disregard them.

How do I use the template? The template consists of the files:

- `clustering_task.ipynb`
- `clustering_algorithms.py`

The Jupyter notebook file `clustering_task.ipynb` is responsible for loading an image, applying clustering functions and presenting the result. The file `clustering_algorithms.py` contains the (incomplete) code for the clustering methods and should be completed with your implementation. In this file, do not change the signature of the methods, only the body. You may add more functions, though. The final file should work with the default `clustering_task.ipynb`. In theory, `clustering_algorithms.py` is the only file you have to edit.

Hint: The relevant parts of `clustering_algorithms.py` have been marked with `TODO` comments.

How many runs do we have to do with k-means? Please use the shown variant of k-means, which runs until there is no more improvement of cluster cost. An arbitrary iteration maximum (e.g. 100) may be used as an additional exit condition.

Which hyperparameters should I use for the clustering? The template contains some values that should work with the provided example image ($minPts = 30$, $\epsilon = 5$ and $k = 4$). These are just suggestions. You may experiment yourself to find good parameters, although this is not part of the task.

Which distance fuction should be used? Euclidean distance would be adequate. It would be ideal to make this function changeable as well, but that is not a requirement.

The template contains a bug! What should I do? In case of errors in the template, we will fix the code as soon as possible. Please report found bugs in the forum so everyone knows about them. We will maintain a list of found bugs and how to deal with them on the Moodle page.

Bibliography

[1] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of the 2nd Int. Conf. on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[2] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.