Lab Class DM:I–II

Until **Wednesday, Feb. 5th, 2020, 23:59 pm CET**, the following exercises must be submitted:

Students receiving 4.5 ECTS: 2, and 3
Students receiving 6 ECTS: 1, 2, and 3

Lab Class General Instructions. Exercises should be completed in groups of three students; which and how many exercises you must submit depends on the number of ECTS credits awarded for the course, based on which degree programme you're studying:

	4.5 ECTS	6 ECTS
Degree programmes	CS4DM, CSM, MI, HCI*	DE, HCI*
	*(HCI students can choose number of credits in electives module)	
For admission to the final exam:		
Must reach at least	80 points	120 points
Out of maximum	120 points	180 points

For each exercise sheet, all required exercises must be submitted until the stated deadline. Each individual exercise will receive a score between zero and ten. Submissions received after the deadline will not be considered for grading. In order to be admitted to the final exams, you must reach the minimum total score shown.
Upload your solutions to Moodle as a single `.pdf` file, plus one `.ipynb` file for any programming exercises. Make it clear which solution corresponds to which exercise. Programming exercises are marked with [P] – their solutions must be documented extensively with docstrings and/or inline comments. ml.weimar.webis.de

Exercise 1 : Hierarchical Cluster Analysis

Consider hierarchical agglomerative clustering of a set of points X in a two-dimensional feature space using the Manhattan distance, i.e. for two points $u, v \in X$, the pointwise distance function is $d(u,v) = \sum_{i=1}^{2} |u_i - v_i|$. We want to explore the conditions under which the two cluster distance measures single link (with $d_{\mathcal{C}}(C, C') = \min_{u \in C, v \in C'} d(u,v)$) and group average link (with $d_{\mathcal{C}} = \frac{1}{|C| \cdot |C'|} \sum_{u \in C, v \in C'} d(u,v)$) merge the points in a different order, and thus result in different dendrograms.

(a) What is the minimal number of points needed in X such that different dendrograms could result, and why? *Hint: consider what happens during the first iterations of the algorithm with small point sets.*

(b) Construct such a minimal point set X, along with the corresponding dendrograms.

Exercise 2 : Hierarchical Cluster Analysis

Given are the following objects from a two-dimensional feature space:

$x_0 = (-5, 4),\ x_1 = (-3, 4),\ x_2 = (-4, 2),\ x_3 = (6, 4),\ x_4 = (8, 4),\ x_5 = (5, 1),\ x_6 = (0, -2),\ x_7 = (2, 0)$

(a) Apply a hierarchical agglomerative clustering algorithm with complete link and Euclidean distance and draw the resulting dendrogram. Note the exact distances on the dendrogram's distance axis.

(b) The dendrogram gives rise to multiple possible clusterings, among them one with two and one with three clusters. Compute the Dunn Index cluster validity measure for both of these clusterings.

Exercise 3 : P Exemplar-based Clustering

(a) Implement the exemplar-based iterative clustering algorithm as a Python function, based on the pseudocode given on the lecture slides, with the following signature:

```
def exemplar_based_clustering(X, k, dist, pick_exemplar, t_max):
    ### [ your code here ... ] ###
```

Your function should expect the following parameters:

- An n-by-p numpy array `X`, containing n points in p dimensions.
- An integer `k` specifying the number of clusters.
- A distance function `dist`, which takes two arguments—each of them a p-dimensional point—and returns a real number.
- A function `pick_exemplar` which takes an m-by-p array (containing all the points in one cluster) as its only argument, and returns a p-dimensional point.
- An integer `t_max` specifying the maximum number of iterations.

Note that this deviates slightly from how the algorithm is specified on the slides. You may use the following "`convergence`" criterion: your implementation should terminate if it finds the exact same clusters in two consecutive iterations. The return value of your function should be a 2-tuple (C, R). The value C is a 1-dimensional array with n elements, each of which is an integer between zero and $k - 1$, where the i-th element of C is your algorithm's cluster assignment for the i-th point in X. The value R is a k-by-p array, containing the k cluster exemplars selected by your algorithm.

Verify that your function can be used to implement the k-means algorithm in the following way:

```
import numpy as np
euclidean_distance = lambda a, b: np.linalg.norm(a - b, ord=2)
pick_centroid = lambda cluster: np.mean(cluster, axis=0)
kmeans = lambda X, k, t_max: exemplar_based_clustering(
    X, k, euclidean_distance, pick_centroid, t_max)
```

Example usage:

```
>>> kmeans(np.array([[0, 0], [1, 1]]), 2, 100)
(array([1, 0]), array([[1, 1], [0, 0]]))
```

Note that your implementation is still correct if the order of the elements in the returned arrays is different.

(b) Implement a function `pick_medoid`. Use this function, as well as `euclidean_distance` and `exemplar_based_clustering`, to implement the k-medoids algorithm.

(c) The following code snippet generates two artificial datasets of two-dimensional points. Run your implementations of k-means and k-medoids on the `blobs` and `moons` dataset, with different values of `k`. Visualize the points in a scatter plot, using different colors for the clusters found by your implementation. Also highlight the cluster exemplars in your plot.

```
import sklearn.datasets as sd
blobs = sd.make_blobs(1000, 2, 3, random_state=42)[0]
moons = sd.make_moons(1000, noise=.05, random_state=42)[0]
```