Problem 6.1 (Hierarchical Clustering) **(25 P.)**
In this exercise, the hierarchical agglomerative clustering algorithm shall be implemented. A description of the algorithm is given in Appendix A.

a) Implement a function that builds the dendrogram for a given dataset using the complete linkage criterion. The function should annotate each tree node with a "size" attribute, where the size is the value of the linkage criterion for merging the two children of the node. (10 P.)

b) Implement a function that creates a clustering of a dataset based on the dendrogram and a maximal value for the size attribute. The function should return the clustering with the minimal number of clusters where each cluster has a "size" value below the given threshold and corresponds to a node of the tree. (10 P.)

c) Apply your method to a dataset consisting of the numbers $\{0, 1, \dots, 9\}$ and the euclidean distance as pairwise element dissimilarity. Create a visualization of the dendrogram and determine the clustering for a maximal size of 4.5. Is the resulting clustering the optimal one or would there be other clusterings with less clusters and sizes below the threshold? (5 P.)

Problem 6.2 (Gradient Descent) **(5 P.)**
Implement the gradient descent algorithm. The algorithm should use at least these parameters:

- The start value x_0,
- the learning rate α,
- and a function that maps x onto $\nabla_x f(x)$.

You could extend the implementation and use the maximum number of iterations to prepare this implementation for problem 6.3. The code scaffold for the algorithm can be found in `gradient_descent.py`. You can test your implementation with the unit test from `test_gradient_descent.py` and debug it with the script `plot_gradient_descent.py`.

Problem 6.3 (Gradient Descent for Linear Regression) **(35 P.)**
Use your gradient descent algorithm to find values $w = (w_0, w_1)$ that minimize $SSE(w)$ on the dataset "`regression_dataset_1.txt`" from Stud.IP for the hypothesis $h_w(x) = w_0 + w_1 \cdot x$ (i.e. $\phi(x) = (x^0, x^1)$). Use the batch update rule, start at $w = (-0.5, 0.0)$, and perform 100 iterations.

a) Implement the functions
   - `predict`, (5 P.)
   - `sse`, (6 P.)
   - and `dSSEdw` (6 P.)

   in the given code scaffold (`linear_regression.py`). The unit tests from `test_linear_regression.py` might help you.

b) For $\alpha \in \{0.0001, 0.001, 0.002, 0.0025\}$: To which value w^* converges the algorithm? What is $SSE(w^*)$? Plot $SSE(w)$ versus the number of iterations $(0, \dots, 100)$ for the different values of α. The code is given in `linear_regression.py`. (3 P.)

c) Based on the results: What can you say about the effect of the learning rate α? Take a look at the convergence behavior and the initial behavior. For example, how can the result for $\alpha = 0.0025$ be explained? (15 P.)

A Hierarchical Agglomerative Clustering

Hierarchical clustering refers to a class of methods in which a dataset is clustered by incrementally merging smaller clusters. Initially, each of the n elements of the dataset is contained in a separate cluster. The algorithm merges in each step those two clusters that minimize a certain linkage value. After $n-1$ merging steps, all elements are contained in a single cluster. The merging process can be seen as constructing a merging tree (the so-called dendrogram). The leaves of this dendrogram correspond to the individual elements and the inner nodes to a cluster containing the elements which are associated to leaves that are contained in the subtree rooted in the specific node.

The algorithm takes as input a function d which returns for two elements their pairwise "dissimilarity". This dissimilarity can be, e.g., the euclidean distance if the elements are real-valued vectors or the dynamic time warping distance if the elements are time series. The linkage criterion defines how dissimilar the elements of a cluster are. In this exercise, we use the "complete" linkage criterion which assigns to a cluster the maximal pairwise dissimilarity of elements in the cluster, i.e., $complete(X) = \max_{x_1, x_2 \in X} d(x_1, x_2)$. Note that the complete linkage dissimilarity can be efficiently updated during the merging process, i.e., when merging two clusters A and B one can use the following formula to compute the distance of the resulting cluster to any other cluster C: $complete(A \cup B \cup C) = max(complete(A \cup C), complete(B \cup C))$.

On the hand-in date, **14.12.2016**, you must hand-in the following: [1]

a) a text file stating how much time you (all together) used to complete this exercise sheet
b) your solutions / answers / code

for problem 6.1 and 6.2 and 6.3.

[1] upload via StudIP (if there are problems with the upload contact me **beforehand**: krell@uni-bremen.de)