Institut für Informatik
Prof. Dr. Heiko Röglin
Magdalena Aretz

Lab Efficient Algorithms for
Selected Problems: Design, Analysis
and Implementation
Winter 2013/2014

Tasklist 3
Due Date: 02.12.2013

Basics

1. **Using Javadoc**:
   Javadoc is a helpful tool to quickly produce a HTML-version of the documentation of your code.

   - Find out how to generate *Javadoc* in Eclipse:
     `http://www.itcsolutions.eu/2010/12/23/tutorial-java-62-2-how-to-generate-javadoc-in-eclipse-or-netbeans/`.
     Note: You will need a JDK (not only a JRE) to use *Javadoc*.
   - Look at the possible paramters of *Javadoc*:
     `http://en.wikipedia.org/wiki/Javadoc`.

2. **GNU R**:

   - If you have not done so already, take your time to work through the introductory material of `R` from the previous tasklist.

Implement

1. Turn your comments into *Javadoc*. You should at least provide one general description of each class and a description of the parameters and return values of the most important methods.

2. `R`: Find out how to perform the following two ways of preprocessing your data:

   - `normalize` – scale each dimension of the data such that it lies in the range [0:1],
   - `standardize` – standardize each dimension of the data such that the mean value is 0 and the variance is 1.
     See `http://en.wikipedia.org/wiki/Standard_score`.

   Compute two new versions of each dataset: the normalized and the standardized one.

3. Java: Make sure you have a function that computes the final error of the `k-Means` algorithm. Remember that this error consists of the summed squared distances of each point to its nearest center:

   $$\Phi(X,C) := \sum_{x\in X} min_{c\in C} \|x-c\|^2,$$

   where X is the dataset and C is the set containing the final centers.

4. Java: Modify your code such that it contains a method *runKMeans* that receives the following parameters:

    - *name* of the dataset
    - a value for k
    - *preprocmode*, which defines which kind of preprocessing is applied
    - *initmode*, which defines how the centers are initialized

    and runs `k-Means` with the specified parameters. You should be able to tell the final error and the number of steps performed until convergence.

5. Java: Write a class *Experiment* that calls *runKMeans* a predefined number of times (e.g. 10 reiterations). It should store one line for each result of running *runKMeans* in a file called *experiments.csv* that has the following format:
    name, n, d, k, preprocmode, initmode, number of steps, final error.

6. Fix one setting (e.g. name = “cloud_01”, k = 10, preprocmode = “standardize”, initmode = “plusplus”) and use *Experiment* to collect the results of at least 20 runs of `k-Means` for this setting.

7. `R`: Find out how to read the file *experiments.csv* and how to select the column you are interested in (e.g. the final error or the number of steps until termination).

    Now generate at least one plot that visualizes the results of your experiment (e.g. you might want to plot the distribution of the final error or the number of steps until termination).