University of Vienna
Faculty of Computer Science
Prof. Claudia Plant
Martin Perdacher

Scientific Data Management

SS 2017

Programming Assignment 3: K-means with LSH

General Remarks:

- This is one of three programming assignments in this lecture. For each assignment you could earn 100 points.
- The deadline is 06.06.2017. No deadline extensions are given.
- If you have problems do not hesitate to contact the tutor or post a question on the Moodle system.
- Pack up your java code, results as a jar-file and the documentation in a .zip file with the following name: group(group number).zip file (e.g. group01.zip) and upload it to the Moodle system.
- It is not allowed to use LSH libraries like *java-LSH*.
- This time you need to include a report on which persons has worked on what tasks and how many hours they spent on them. Each team member has to sign it.

Task 3-1 Implement LSH for K-Means

External resources:

- NMI implementation in Java:
  `https://gist.github.com/perdacherMartin/76689fdf2c950fbeba6b013d09906de4`

In this programming assignment you will implement Locality Sensitive Hashing (LSH) for K-means on a synthetic dataset *LSH-nmi.csv*, which you can download on the moodle platform. This is a 10 dimensional dataset with approximately 290.000 data points and 15 cluster ($k = 15$). Write a Java program which implements LSH for K-means, as introduced in the lecture slides. There is also a note, that you should combine your hash-values with AND/OR to reduce false negatives and false positives. The goal of applying LSH to K-means is, that we speed up the calculation of K-means, by reducing the cost of expensive distance calculation. Here we try to find out how much gain in performance and quality we have by adding LSH to our K-means implementation.

We know the ground truth to our dataset, to which we will compare your K-means result (last column in our dataset). Compare your result with these numbers using the NMI implementation provided in Java. The NMI value ranges between 0 and 1. The closer the value to 1, the better your clustering result.

- Preprocess the dataset, to make it suitable for the NMI calculation.
- Write a Java program, which uses LSH to speed up the distance caluclation. Try out different settings, where you combine different hash functions with AND and OR as it was discussed in the lecture. In example you can have one setting where you combine two hash functions with AND and in the second setting you combine two functions with OR, but you could try out several combinations, with different hash functions. Try also to vary the number of buckets of your hashing function. Measure the runtime $time_{LSH}$ of your LSH implementation.
- Compare the runtime of the two different settings with the K-means implementation provided by us. It is a solution by one of our students for the first assignment. Measure the time $time_{ref}$ in a fair setting.
- Select the best two settings and provide the NMI and the speedup value s, where $s = \frac{time_{ref}}{time_{LSH}}$.
- Battle Royal. We will give bonus points for the best solutions. We will choose the best solutions with a skyline query. Given a set of points $p_1, p_2, ..., p_N$, the skyline query returns a set of points P (referred to as the skyline points), such that any point $p_i \in P$ is not dominated by any other point in the dataset (for more details see `http://www.cs.umd.edu/class/spring2005/cmsc828s/slides/skyline.pdf`).
- Visualize your results.
- Discuss your results.