Databases Project Status Report

Thai Flowers

Goal: Implement RNN-DBSCAN and reproduce experiments in the original paper

Implementation

For this project I choose to use the Java programming language out of a desire to avoid debugging pointer errors, access to a large number of libraries, and the chance to use IntelliJ IDE. I choose to use the "Apache Commons Math" library out of familiarity, because it ships with an implementation of DBSCAN and the belief it's clustering framework / class-hierarchy would serve as a guiding basis for implementing the DBSCAN frameworks. Due to issues with documentation I moved to the "Hipparchus" fork of the library. In order to perform k-nearest-neighbor (kNN) search and reverse kNN search I found the "java-graphs" library, which implements various algorithms for creating a kNN graph and implemented a transpose function for creating a reverse kNN graph from it. I also leveraged the "JFreeChart" for displaying clusters found in 2d data sets.

Testing on the t4.8k dataset revealed significantly poor execution time performance against vanilla dbscan. A more thorough perusal of the paper revealed that the RNNDescent algorithm was used and provided the parameters found to work the best in their experiments. Some informal experiments revealed that java-graph's implementation of RNNDescent gives poor results with the paper's parameters, so more stringent parameters were used (namely 12 iterations are recommended but 15 at a minimum is effective). This results in run times only a few seconds slower than vanilla DBSCAN.

Leveraging the RNN-DBSCAN implementation I quickly implemented ISB-DBSCAN and RECORD using the same techniques. Though I believe ISB-DBSCAN may be implemented incorrectly as it gives extremely poor results.

Datasets: Most of the datasets are freely available at http://cs.joensuu.fi/sipu/datasets/ and at the UCI Machine Learning Repository. I have downloaded all the datasets available at the former web site, and produced a shell script to translate datasets in the Shapes collection into the format given in t4.8k that my programs operate on.

I still need to use scikit-learn or similar to recreate the custom artificial datasets used in the paper, namely Moon, Blob, Circle, Swiss Roll, and (most importantly) Grid. The last of which is used to test explicitly the effectiveness of the density calculations.

Next steps: Create the missing datasets, implement IS-DBSCAN and OPTICS, run the parameter testing experiments in the paper, followed by performance experiments. If I have time I'll experiment with replacing the Java-Graphs implementation of RNNDescent with a custom data structure based implementation (perhaps kd-trees).