# CSCI 4372/6397: Data Clustering

## Phase 2: Normalization and Initialization

Submission Deadline: March 19 (23:59:59)

**Objective:** Normalize your data and implement an alternative initialization method for k-means.

**Part I – Normalization**: In clustering applications, attribute normalization is a common preprocessing step that is necessary to prevent attributes with large ranges from dominating the distance calculations and to avoid numerical instabilities in the computations. In this phase, you will implement the "min-max normalization" method to normalize your attributes prior to clustering. You can easily find the description of this simple normalization method in any Data Mining book, or on the WWW, for example, from here
https://www3.nd.edu/~rjohns15/cse40647.sp14/www/content/lectures/07%20-%20Data%20Transformation.pdf

**Bonus for Undergraduate Students [10 points] / Mandatory for Graduate Students**: In addition to "min-max normalization", implement the "z-score normalization" method.

**Part II - Initialization**: In Phase 1, you implemented the standard batch k-means algorithm. Unfortunately, k-means is highly sensitive to initialization. In other words, different initial centers can result in vastly different results. Numerous initialization methods have been proposed to address this problem. In this phase, you will implement another initialization method for k-means and compare it with the "random selection" method that you implemented in Phase 1. You will compare the two initialization methods with respect to the following factors:

- **Initial SSE**: This is the SSE value calculated after the initialization phase, before the clustering phase. It gives us a measure of the effectiveness of an initialization method by itself.
- **Final SSE**: This is the SSE value calculated after the clustering phase. It gives us a measure of the effectiveness of an initialization method when its output is refined by k-means. Note that this is the objective function of the k-means algorithm.
- **Number of Iterations**: This is the number of iterations that k-means takes until reaching convergence when initialized by a particular initialization method. It is an efficiency measure independent of programming language, implementation style, compiler, and CPU architecture.

In Phase 1, you implemented the "random selection" method. In this phase, you will implement the "random partition" method. In this method, starting from empty clusters, first assign each point to a randomly selected cluster. You then take the centroids of these initial clusters as the initial centers.

**Bonus for Undergraduate Students [10 points] / Mandatory for Graduate Students**: In addition, to "random partition", implement the "maximin method". This method chooses the first center arbitrarily from the data points and the remaining (K − 1) centers are chosen successively as follows. In iteration i (i = 2, 3,…, K), the i-th center is chosen to be the point with the greatest Euclidean distance to the nearest previously selected (i − 1) centers. In other words, center 2 is chosen to be the farthest point to center 1. Center 3 is chosen to be the point with the greatest distance to the nearer of centers 1 and 2, that is, point **x** with the greatest min (d(**x**, $\mathbf{c}_1$), d(**x**, $\mathbf{c}_2$)) value, where $\mathbf{c}_1$ and $\mathbf{c}_2$ are the first, and second centers, respectively, 'd' is the Euclidean distance, and the function "min (a, b)" returns the smaller of 'a' and 'b'. Center i is chosen to be the point **x** with the greatest min (d(**x**, $\mathbf{c}_1$), d(**x**, $\mathbf{c}_2$), …, d(**x**, $\mathbf{c}_{i-1}$)) value. Note that maximin can be implemented in O(NDK) time, where N, D, and K respectively denote the numbers of points, attributes, and clusters. By contrast, a naïve implementation requires O($N^2$DK) time. If you go for a naïve implementation, you are likely to have difficulty completing your runs.

If your initialization method is randomized, execute it <R> times and tabulate the best result for each performance measurement separately (best initial SSEs, best final SSEs, and best # iterations) for each data set. Otherwise, a simple table of the data sets and the corresponding measurements will be sufficient. There are many ways to tabulate this kind of data; try to find an intuitive way.

**Output:** Microsoft Excel or Apache OpenOffice Calc table(s) of the performance measurements for each combination of normalization and initialization methods. In other words, {10 data sets} x {1 or 2 normalization methods} x {2 or 3 initialization methods} x {3 performance measures} = 60 or 180 values. Of course, you can split this table into smaller tables. For example, you can put 6 or 18 values into one table that represents a particular data set.

**Language:** C, C++, or Java. You may only use the built-in facilities of these languages. In other words, you may not use any external libraries or APIs.

**Submission**: Submit your source code and output file(s) via Blackboard. Do **not** submit your files individually; pack them in a single archive (e.g., zip) and submit the archive file.