CSCI 4372/5372/6397: Data Clustering

Phase 3: Internal Validation

Submission Deadline: **November 15 (23:59:59)**

Objective: Implement internal validation methods to determine the number of clusters automatically.

Perhaps the most significant drawback of k-means is that it requires that the number of clusters (K) be supplied by the user. In many applications, this is either impossible or impractical.

In this phase, you will implement two internal validation methods that will help k-means automatically determine the number of clusters. An internal validation method quantifies the match between an automatically generated partition of a data set and the data set itself. Internal validation is often accomplished using an internal validity index, a function that takes a partition, the data set itself, and possibly some additional parameters as input and gives a numerical value indicating the quality of the partition as output.

In this phase, you will implement the Calinski–Harabasz (CH) and Silhouette Width (SW) internal validity indices. These indices are described in many resources, see for example, the following recent book by Zaki and Meira Jr.: https://dataminingbook.info/book_html/

Bonus for 4372 Students [10 points]; Mandatory for 5372 and 6397 Students: In addition to CH and SW indices, implement the Dunn (D) index (for a description, refer to the aforementioned book).

Bonus for 4372 and 5372 Students [10 points]; Mandatory for 6397 Students: In addition to CH, SW, and D indices, implement the Davies–Bouldin (DB) index (for a description, refer to the aforementioned book).

CH, SW, and D are maximization indices, whereas DB is a minimization index. The way these indices are used is quite simple. For a given data set, we first decide K_{min} and K_{max}, the minimum and maximum number of clusters that might be present in the data set, respectively. For example, for the iris data set, K_{min} and K_{max} could respectively be 2 and 9. For any data set, K_{min} is typically 2, whereas, a rule of thumb on the maximum possible K_{max} value is the nearest integer to \sqrt{N/2}, where N is the number points in the data set. Assuming that we have a randomized initialization method, we then run k-means R times for K = 2 and compute the internal validity index (say, CH) on the partition generated in the best run (that is, the run that produced the smallest SSE). We do the same for K = 3, K = 4, …, K = 8, and, finally, K = 9. Since, CH is a maximization criterion, we then find the K value that produced the largest CH value. That K value is the "estimated" number of clusters in the data set. For a numerical example, refer to the example given below and to Example 17.8 in the aforementioned book. Note that, these indices all give *estimates*; so, you may not find the "correct" (3) number of clusters for iris (or, for any other data set).

Numerical Example for SW: SW values for the iris data set (only for K = 2 and 3 clusters—in your experiments, your K_{max} value for this data set should be \round{\sqrt{150/2}} = 9) are given below (obj = SSE; delta obj = relative reduction in SSE between a particular iteration and the iteration preceding that). Note that this is just to give you an example of how reasonable SW values should look like; you may **not** get the exact same numbers depending on your initialization. In addition, these numbers are obtained on the raw (unnormalized) iris data set. In your experiments, you should normalize the data sets using the min-max method (Phase 2). There are 3 real clusters in iris, but two of those clusters overlap, so K = 2 gives a much better SW value than K = 3. So, if we were to try just K = 2 and 3 clusters, based on the SW index, we would select K = 2 as the best estimate for the number of clusters in this particular data set. Keep in mind that if you are getting silhouette values outside the [-1, 1] range, you must be doing something wrong.

./test data/iris_bezdek.txt 2
Iteration 1: obj = 539.413 ; delta obj = 0
Iteration 2: obj = 366.075 ; delta obj = 0.321347
Iteration 3: obj = 237.899 ; delta obj = 0.350135
Iteration 4: obj = 175.767 ; delta obj = 0.261168
Iteration 5: obj = 154.339 ; delta obj = 0.121915
Iteration 6: obj = 152.513 ; delta obj = 0.0118286
Iteration 7: obj = 152.348 ; delta obj = 0.00108328
SW(2) = 0.850351

./test data/iris_bezdek.txt 3
Iteration 1: obj = 230.163 ; delta obj = 0
Iteration 2: obj = 81.9806 ; delta obj = 0.643815
Iteration 3: obj = 79.3943 ; delta obj = 0.0315479
Iteration 4: obj = 78.9101 ; delta obj = 0.00609958
Iteration 5: obj = 78.8514 ; delta obj = 0.000742812
SW(3) = 0.73566

Hint: The trace of the within-clusters scatter matrix, $tr(S_w)$, is the same as SSE, which is already calculated by k-means. Therefore, you do **not** need to calculate $tr(S_w)$ separately. Also, for $tr(S_b)$, you do **not** need to calculate the entire S_b matrix. Trace of a square matrix is given by the sum of its diagonal elements. Hence, all you need to do is to calculate the diagonal elements of S_b and then add them up.

Output: Microsoft Excel or Apache OpenOffice Calc table(s) of each internal validity index for various K values for each data set. For each data set and validity index, show the estimated number of clusters in **bold**. **Discuss** which index seems to estimate the number of clusters more accurately. For initialization, 4372/5372 students **must** use the "random partition" (Phase 2) method (R = 100), whereas 6397 students **must** use the "maximin" (Phase 2) method (R = 100). All data sets **must** be normalized using the "min-max" (Phase 2) method.

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.

Submission: Submit your source code and output file(s) through Blackboard. Do **not** submit your entire project folder or the input data files that I provided to you (I already have them 😊). In addition, do **not** submit your files individually; pack them in a single archive (*e.g.*, zip) and submit the archive file.

Grading: *Functional correctness* and *adherence to good programming practices* are respectively worth **90%** and **10%** of your grade. Given a valid input, a functionally correct program is one that produces the correct output. There are a lot of generic or language-specific guides on good programming practices on the WWW. You should identify an appropriate one, include its URL at the top of your program source code(s), and use it throughout this project. In general, you should pay more attention to structural issues (*e.g.*, modularity) than cosmetic ones (*e.g.*, naming and formatting). Keep in mind that if your program is incorrect, whether or not you adhered to good programming practices is immaterial.