Programming Assignment: Clustering Validation

Problem Description

In this assignment, we will be implementing two clustering validation measures: Normalized Mutual Information (NMI) and Jaccard similarity. To help with the implementation, we will first implement the confusion matrix calculation, where each cell $C_{i,j}$ in the matrix is the count of the number of examples with true label i and predicted label j.

Your task: complete the missing functions in the given programming template files.

Programming Template

Find in `template/`

The templates consist of three methods to calculate the confusion matrix, NMI, and Jaccard similarity. The argument inputs to all three methods are integer lists (Python), vectors (C++) or arrays (Java) consisting of the true labels and the predicted labels.

You are allowed to use *one of three* programming languages for this assignment: Python (3.10), C++ (14), or Java (JDK version 17). You are provided with the template code for each language containing descriptions for the expected input and output for each function that you would have to write.

Output Format

The output format for the NMI and Jaccard similarity methods are floating point numbers. **Please do NOT round your results**. The output format for the confusion matrix method is a dictionary (Python) / map (Java) / unordered_map (C++).

Sample Input and Output

Find in `sample_test_cases/`

To aid your understanding, we have included one test case for each of confusion matrix, Jaccard, and NMI. Each input file has $N + 1$ lines. The first line is a single integer that represents test_id. test_id is 0, 1, 2 for Jaccard, NMI, and confusion matrix. For example, if the first line is 0, it means this test case is a test case for Jaccard. The following N lines have two integers each. The first integer is a true cluster label. The second integer is a predicted cluster label.

For Jaccard and NMI tests, the output file has a single float number, rounded to precisely 4 decimal points. (**The rounding is for debugging only. Do NOT round your results when submitting**).

For confusion matrix tests, the output file has multiple lines. Each line has three numbers i, j, v, representing $C_{i,j} = v$ in the confusion matrix. The lines are sorted by coordinates (i, j).

Note here that the test case format is provided for you to understand the test case files that will be used. For whichever language you use, you would NOT have to read in test cases from STDIN. That will be handled by the grader. You simply need to complete the required functions in the template code file.

Test Scale

For all public and hidden test cases, $N \leq 100$, where N is the number of data points.

Allowed Libraries

For each language, only standard built-in libraries would be usable. For example, if using Python, you would NOT be able to use libraries like NumPy or Scikit-Learn.

Submission

You need to submit ONE completed template code file to Gradescope. `submission.py` or `submission.cpp` or `Submission.java`. Note the capitalization for the `.java` file. For C++, please do not submit the submission.h file to Gradescope.

On Gradescope, your code will be tested on a list of public test cases, for which, you will be able to view both the input and the expected output. At the end of the submission deadline, you will be able to see the results produced by your code on a list of hidden test cases.