Project 1 (Part 1) – Agglomerative Clustering

Handed out: Tuesday, 11/15/2016	Due date: Tuesday, 11/22/2016

Submission instructions

Inside that the homework subdirectory, create a directory for homework #6, and call it `lab6`. When you finish the assignment, go to the homework directory and submit it as follows:

```
$ submit movery project1_1 project1_1
```

Problem

Your assignment is to implement a agglomerative clustering algorithm for real-valued vectors in multidimensional space.

Terminology & Definitions

Clustering is a task that involves grouping similar objects together in such a way that objects in the same group (aka cluster) are more similar to each other than to objects in other groups (clusters).

Agglomerative clustering is a method for producing a hierarchical tree of clusters in a process that successively merges clusters with each other to produce larger clusters.

- The process begins with a set of objects that need to be clustered with each object forming a separate cluster.
- Pairwise distances between all clusters are computed, and the two closest clusters are merged. The process is then repeated.
- This process results in a series of merges which eventually result in a single cluster at the top of the cluster tree that contains the full set of objects.

In order to implement agglomerative clustering, we need to define two distance measures:

- The distance between the objects that are being clustered.
- The distance between the clusters.

Computing Distance

The distance metric between two vectors **p** and **q** is computed as the standard Euclidean distance:

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2+(q_2-p_2)^2+\cdots+(q_n-p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}.$$

The distance between two clusters should be computed as the distance between their centroid vectors, where a centroid vector is computed by taking an average for each coordinate over all vectors in the cluster, e.g. given a set of vectors in 4-dimensional space

$\boldsymbol{p} = \langle p_1, p_2, p_3, p_4 \rangle$

$\boldsymbol{q} = \langle q_1, q_2, q_3, q_4 \rangle$

$\boldsymbol{r} = \langle r_1, r_2, r_3, r_4 \rangle$

- their centroid would be computed as

$centroid(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}) = \langle (p_1 + q_1 + r_1)/3, (p_2 + q_2 + r_2)/3, (p_3 + q_3 + r_3)/3, (p_4 + q_4 + r_4)/3 \rangle$

Hint: If you begin with a cluster that contains a single vector and successively add vectors to it one by one, the centroid computation is simplified if you keep a running sum for each coordinate over all cluster elements, as well as the total number of elements in the cluster. The centroid for the updated cluster can then be computed by dividing the running sum for each coordinate by the number of elements in the cluster.

Hint: You can make your implementation faster if you don't recompute similarity between individual points / clusters.

Specification

Your program should

- Read a set of real-valued vectors from standard input
- Take the number of clusters K as command-line argument
- Cluster the input vectors into K clusters
- Print out the resulting set of clusters to standard output

Dimensionality of the vectors should not be specified in advance, but you do need to check that all the input vectors have the same number of dimensions.

Your program should assume the following usage:

```
$ cat input.txt | project1_1 [number-of-clusters] > output.txt
```

- where the number of clusters is an integer value.

Please print an error message and exit if

- Command-line argument is not an integer.
- The number of dimensions varies across different vectors read from the input file.

Example

For example, given the following invocation and input, your program should produce the following output:

```
$ cat input.txt | project1_1 2 > output.txt
```

Input file `input.txt`:

```
10.1 23.1 4.9 56.84
1.2 23.4 4.2 5.35
2.5 67.4 5.6 34.9
11.32 2.1 4.12 15.3
```

Output file `output.txt`:

```
cluster #  1
11.32, 2.1, 4.12, 15.3
1.2, 23.4, 4.2, 5.35
10.1, 23.1, 4.9, 56.84

cluster # 2
2.5 67.4 5.6 34.9
```

Grading Criteria

The spec for this assignment contains 4 requirements:

1. Read vectors from stdin
2. Receive K from command line
3. Generate K clusters
4. Print to stdout

This assignment will be graded as follows: 15 points for generating clusters, 5 points combined for the other requirements. Extra points will be given for caring about performance (e.g. caching the results of similarity computation, etc.)