PA6: Clustering

This programming assignment is to be completed in either Java or Python (your choice) and submitted on Gradescope.

Collaboration is restricted to just 2 people (other than yourself), and it must be "Whiteboard Only". In summary, this means that you may discuss the assignment with 2 others, but you cannot preserve any artifacts (digital or physical) from these discussions. In particular, this means you may not take and keep photos, notes, video, audio recordings, whiteboard contents, etc. from any discussions.

All collaborators and other resources consulted should be listed as a comment at the top of your code submission. Course staff and official course materials do not need to be listed.

Motivation

There are two primary categories of machine learning tactics. Supervised learning involves using labelled data to train a model to later apply labels to new data. For example, suppose I had a bunch of photos of Labradors (a dog breed) that I wanted to separate into black labs, yellow labs, and chocolate labs (different varieties of Labradors different only by coat color). If I manually pre-labelled these photos with the correct variety then I could use them to train a supervised learning algorithm to label future photos with the correct variety of Labrador.

An unsupervised model, on the other hand, does not require any pre-labelling of the training data. Instead, unsupervised models attempt to find patterns in the training data. For example, if I photos of Labradors we might be able to plot these photos as points in some cartesian space (as a simple example, we might have a 3-dimensional point per picture which represents the RGB of the coat color). With these points, an unsupervised model might identify clusters of points that are similar to one another

and conclude that points in the same cluster must be the same Labrador variety. The advantage to these "clustering" algorithms is that we do not need to take the effort to pre-label, one disadvantage is that it does not tell us which cluster represents which variety. There are other advantages and disadvantages as well, but this is sufficient for now.

Problem Description

For this problem set we will write a clustering algorithm. Consider that we have n items in our data set. The input to your algorithm is going to be a $n \times n$ array of distances (cell i, j of this array represents the distance from item i to item j, we will guarantee that cell i, j is equal to j, i and each cell i, i is 0, so you may think of this as an undirected, weighted, complete graph), and an integer k. This clustering algorithm will form the items into k clusters to maximize the smallest distance between the closest pair of items that span any two clusters. The intuition being that this maximizes the distance between the closest two clusters.

For example, suppose we have 5 items that we would like to cluster into 3 clusters, and the item distances are those in the 2-d array below.

```
 0  18  21  23   5
18   0  54  30  31
21  54   0  15  32
23  30  15   0  15
 5  31  32  15   0
```

In this case the optimal clustering would be for the first cluster to contain items 0 and 4, the second to contain items 2 and 3, and the third to contain item 1. To see the cost of this clustering we look at the closes pair of items for each pair of clusters, and then save the smallest. So in this case the cost of the clustering would be 15. The following explains why:

– The distance between the first cluster and the second is 15 because the closest pair of points that crosses these clusters is items 4 and 3 which have distance 15.

– The distance between the first cluster and the third is 18 because the closest pair of points that crosses these clusters is items 0 and 1 which have distance 18.

– The distance between the second cluster and the third is 30 because the closest pair of points that crosses these clusters is items 1 and 3 which have distance 30.

– The cost of the clustering is the smallest of these distances, and so it is 15.

The goal of the clustering is to maximize this cost.

Input: A number k indicating the number of clusters to break the items into and an $n \times n$ array of floats/doubles which indicates the pairwise distances of all items.

Output: Your algorithm should print just the cost of the clustering, so in the above example your algorithm should print 15 or 15.0 to the console.

Input Format

Input will be given in the console (same as with the previous PAs). To provide an input with n itmes, you will have $n + 2$ lines in your input as follows:

– The first line contains the value of k, i.e. the number of clusters to break the items into.

– The second line contains the value of n, i.e. the number of items.

– The remaining n lines contain a space-separated list of floats/doubles which indicate the distances between each pair of items

For the example above, the input would therefor look like:

```
3
5
0 18 21 23 5
18 0 54 30 31
21 54 0 15 32
23 30 15 0 15
5 31 32 15 0
```

Starter Code Summary

The starter code provided consists of two files:

– `InputReader.java/input_reader.py`: This is the program you will use to run your code. You will not submit this program, instead the autograder will be using its own version to interact with your code. This program simply reads the input file and then invokes the function you will implement.

– `Clusterer.java/clusterer.py`: This is the program that you will edit and submit. There is just one function in the program. This function, called `cluster_cost` takes in the value k (the number of clusters) and the 2-d array of item distances. It then prints the cost of the clustering to the console.

Hopefully Helpful Hints

To help you to get started with thinking about this problem, keep these in mind:

– If we consider the points and their distances as a weighted and undirected graph then the distance that ends up representing the final cost must be an edge in a minimum spanning tree of the graph. Suppose that the cost of the clustering is the weight of edge (x, y). We will define a cut in the graph such that the cluster which contains x is on one side of the cut and all other nodes are on the other side. In this case, any edge going from a node in x's cluster to a node in the second must cross the cut. The closest pair must then be the lightest edge which crosses the cut, and therefore is part of a minimum spanning tree.

– Being a spanning tree, a minimum spanning tree is connected and acyclic and contains $n - 1$ edges. If any edge is removed then the graph is no longer connected, instead it will have two separate components. If any two edges are removed then it will have three separate components.