CSCI598C: Final Project
UPGMA Report

Allee Zarrini

December, 2019

Algorithm Description

The UPGMA algorithm constructs a phylogenetic tree for a group of protein sequences given a distance matrix. Phylogenetic trees show the evolutionary relationship among organisms. A distance matrix shows the distance between protein sequences. A smaller distance between proteins means that the sequences are more similar. More similar sequences imply that the sequences share a common ancestor.

The algorithm begins by creating a cluster for every sequence. Then it searches for the smallest distance between 2 clusters. Once found, it creates a new cluster which is the combination of the two found clusters. Then the algorithm computes distances for the new cluster with all the remaining clusters. This is done until there are only two clusters remaining. In which case the final two clusters are joined together.

Algorithm Implementation and Verification

I implemented UPGMA using C++ on macOS Catalina. The C++ is compiled using the clang compiler. I've modified the code to be able to run on the alamode machines and gcc. To run the code, simply run the script "run-tests". This script will compile my code and generate 10 random distance matrices of varying sizes. The matrices passed as input into my executable C++ program "upgma". The output of my program is compared to a python library called "biopython".

This library contains its own version of the UPGMA algorithm and this is our method of verification. This library also has helpful functions to print out our output in a tree format. To get access to this library, simply use "pip install biopython".

A series of tests will run and they will display a tree generated from my output, and one from "biopython". The trees will not look identical in structure but all the pairings for the sequences will be the same. The only time I have noticed incorrect results from my code is in the case where there are two distances in our matrix that are the same. In this case, depending on which pair is chosen by the algorithms, the results may vary. However, this problem does not occur in our test cases as I am generating random doubles instead of integers. Besides, this does not imply incorrectness of the algorithm.

Each component and script can be run individually as well. To compile C++ files, simply run "make" on the command line. I used python2 to run my python script. Further instruction on the use of command-line arguments are given when the scripts are run.

Result Analysis

For every test that I've ran on my personal machine, I've observed correct output from my implementation.