Lab 3 - Parallelizing k-means Stat 215A, Fall 2017

Aummul Baneen Manasawala

October 23, 2018

Optimizing the Time Complexity of Similarity Matrix

Amongst the three alternative measures of similarity mentioned in the paper, we chose the correlation or cosine similarity measure which was advocated by Fowlkes and Mallows. We justify the choice by its simplicity and geometric interpretability as well as strong algebriac intuition prevalant in the masses. The calculation involved the for loops on matrix which was computationally heavy. This led us to investigate the methods that could save time and resources in computation.

The for loops in R take much longer to run than the for loops in C++. This is because of the fact that every operation carries around a lot of extra baggage. Therefore, we considered using the C++ code to form the similarity matrix rather than the base R code as we had to form a hugh matrix that would be in the order of 30000*30000. Inorder to prove our hypothesis that the C++ code would significantly reduce the time and the computation required to calculate the similarity, we did microbenchmarking.

Although the similarity coefficient obtained from both the languages is exactly the same, but the time characteristics have striking differences. In order to study them in detail, we plot the distribution of the time required for the computation in R and C++ (figure 1). We find that R is not only significantly slower than C++, but also it's distribution of time required has a light tail and hence more uncertainity associated with it than the heavy tailed distribution of the C++ code. To put things in perspective, on average, R takes approximately 100 times more time than C++ code. Proving our earlier hypothesis from this analysis, we chose to run our algorithm to choose the number of clusters in the data from the stability perspective using the similarity computed using C++ code that could handle the for loops much more efficiently than R.

Avoiding matrix storage

The space complexity of the similarity computation for 0.8 fraction of the data is 29,000 * 29,000. Thus, a method that could by-pass the storage of labeling matrix could be benificial significantly for the well being of the memory of our computer. Therefore, we devised a method that did not need matrix storage for the similarity calculation. This was achieved by simultaneously adding the the dot product of each element in a variable instead of forming a matrix that would be multiplied at the end.

Application of the stability algorithm for discovering structure of Linguistic Variation

We apply the model explorer algorithm suggested by Ben Hur et al. in their paper ”A stability based method for discovering structure in clustered data”. This computation was optimized ny using the parallelization of the outerloop that covered the number of clusters into 4 cores.

Choosing the number of clusters

Using the figure 2, we observe that at k=3, the histogram is concentrated at 1 since almost all the runs form the same clusters. None of the other values of the number of cluster is as concentrated towards one as when

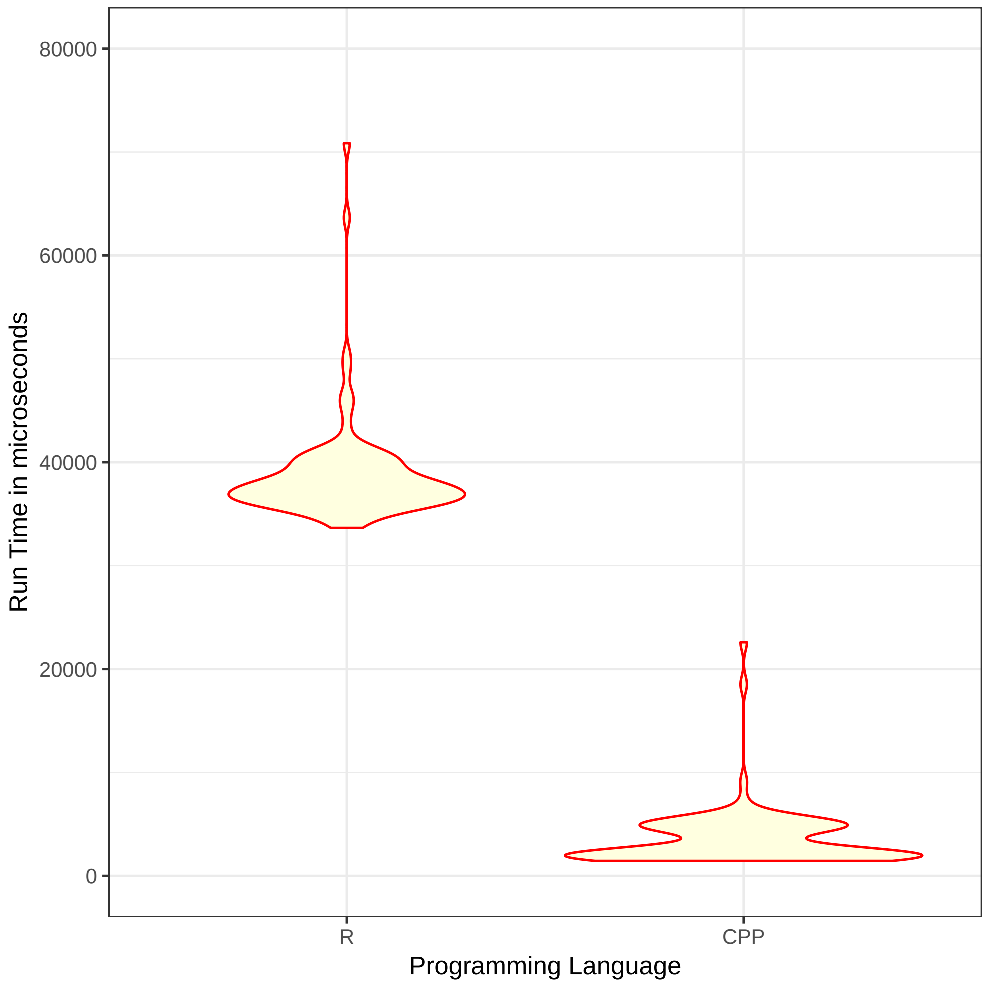

Figure 1: Comparision between the computation time for the similarity co-efficient from R code and C++ code

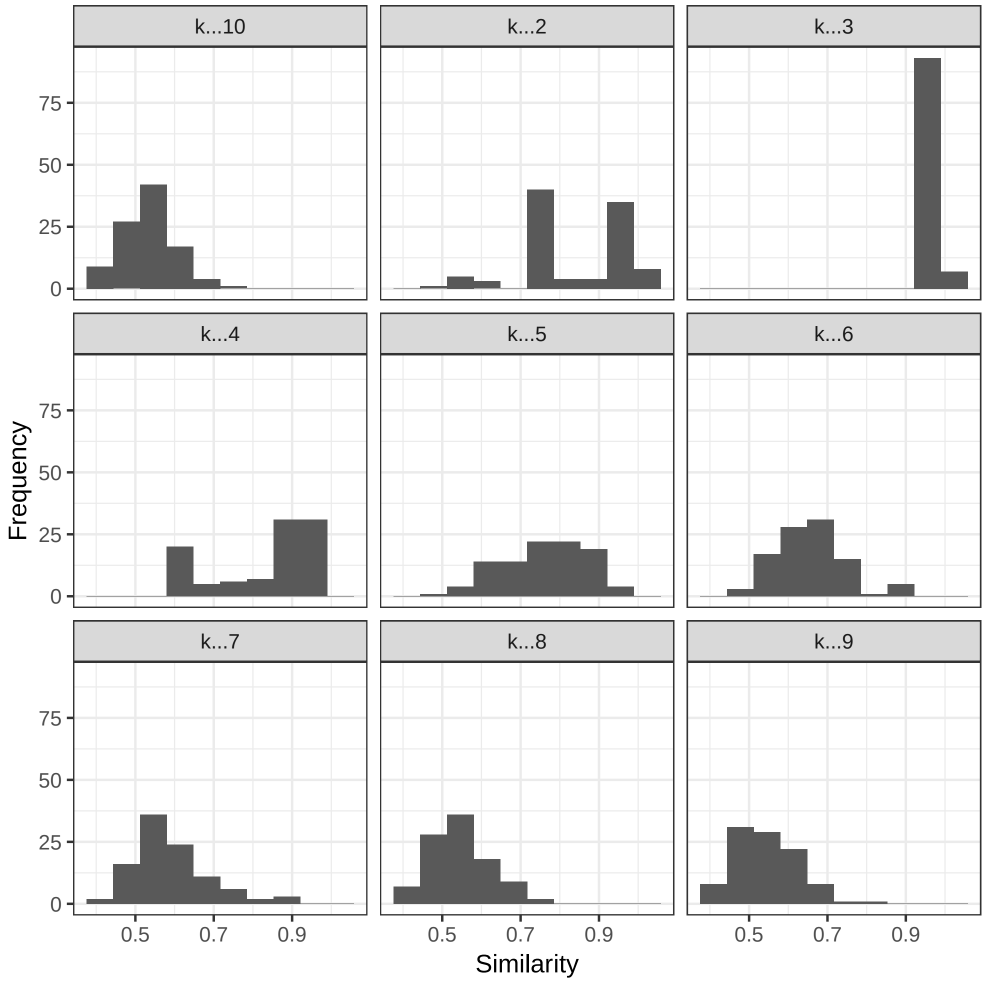

Figure 2: Histogram of the correlation similarity measure for various number of clusters

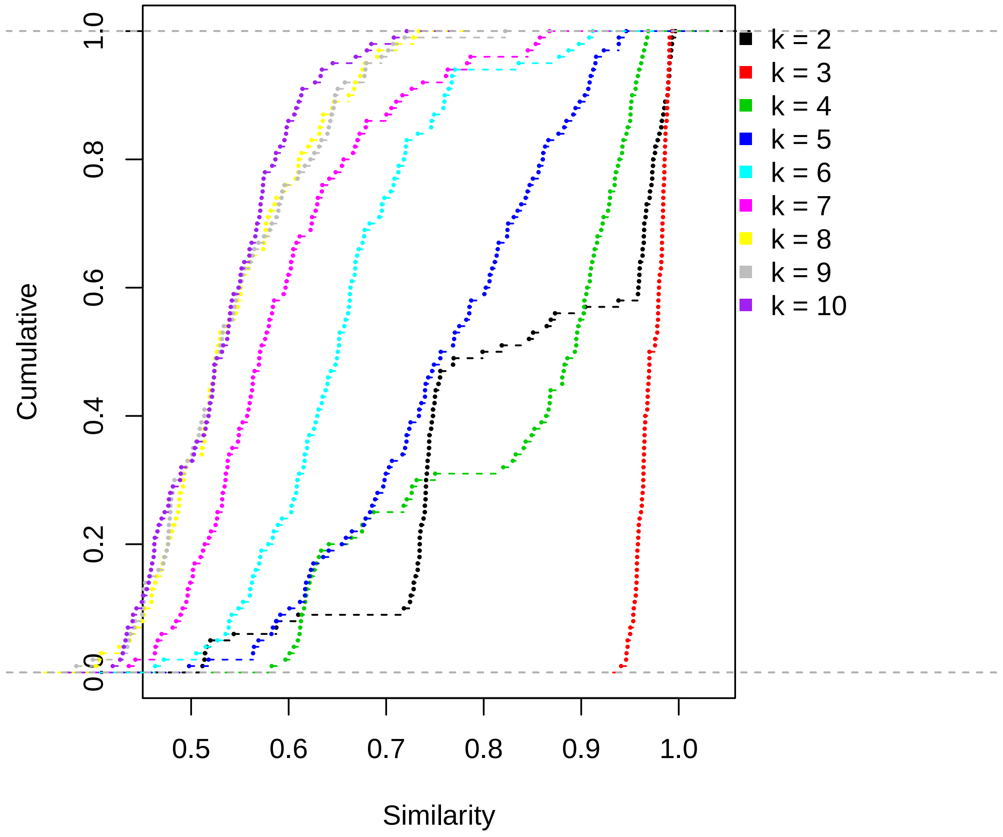

Figure 3: Cumulative distributions for increasing values of k

k is 3. This suggests that no matter however much you purturb the data, the clustering unanimously makes three same clusters in the data. Therefore, we choose the number of clusters to be **3**. We can also observe from the figure 3 that the number cumulative distribution when the data is clustered into 3 is optimally close to 1 in similarity to further bolster our choice.

Discussion on the algorithm

Since this algorithm perturbs the data by subsampling large number of times and checks the similarity of each of the sub samples for each case of possible number of clusters, I find it a pretty robust as well as trustable method to choose the number of clusters. I feel that this method could be coupled with the elbow curve to strengthen the choice. A deep and prominent elbow would also be robust and stable with the changing intialization and perturbations in the data. This would not be computationally as expesive as the algorithm suggested by Ben Hurr et al. in the paper. So, we must consider it if the elbow is not very prominent to make the decision about number of clusters.