Lab 3 - Parallelizing k-means Stat 215A, Fall 2018

Todd Faulkenberry

20 October 2018

1 Introduction

In this project, we will implement and analyze an algorithm for computational stability that was pioneered by Ben-Hur et. al. (2002). During our analysis, we will use the algorithm to see its results when conducting k-means 100 times for each k between 2 and 10. We will end the paper with a more general discussion of the algorithm and if it seems trustworthy after our analysis.

2 The Algorithm: Implementation and Results

While the paper mentioned a handful of similarity measures to use, I chose correlation because it was the measure with which I had the most familiarity. Because this algorithm is so computationally intensive, we were expected to parallelize it on the SCF cluster. I intended to do this with an m of .5 – meaning each subsample for k-means would have 50% of the overall observations – but I had trouble getting the SCF cluster to recognize my data after I loaded it onto the server. I attended office hours to try to remedy this, but even with the help of the GSI and a few other people there, I could not fix it. As such, I parallelized the k-means on my machine. In order to save time, I lowered m from .5 and .2, so each subsample in my analysis only has 20% of the observation. In effect, this changed the question of interest for my project from "how effective is this stability algorithm?" to "how effective is this stability algorithm at smaller subsample sizes?"

As Ben-Hur discusses in the paper, the algorithm reveals which k is the most stable through seeing which k consistently received a high similarity measure – in this paper, correlation. By running k-means on two separate subsamples and then taking the correlation of the clusters of those subsamples, we can see if the same data points are appearing in the same clusters even when they appear in different subsamples. After doing this process many times over – in this case, 100 times per k – we can get a sense of which k is most computational stable, i.e. which k is consistently assigning data points into the same clusters Ben-Hur et. al. (2002).

To this end, Ben-Hur describes a good k as one that does not have a wide spectrum of possibilities, and where the majority of calculated correlations are close to 1. At lower ks (i.e. 1 or 2), this can sometimes be misleading as there are very few clusters from which to choose and thus data points may end up in the same clusters by default. As k increases, however, we will begin to see points, on average, move away from 1 as a the spread widens. Thus, the correct k if the algorithm works will 1) have most data points near one, and 2) not offer a huge range of correlations Ben-Hur et. al. (2002).

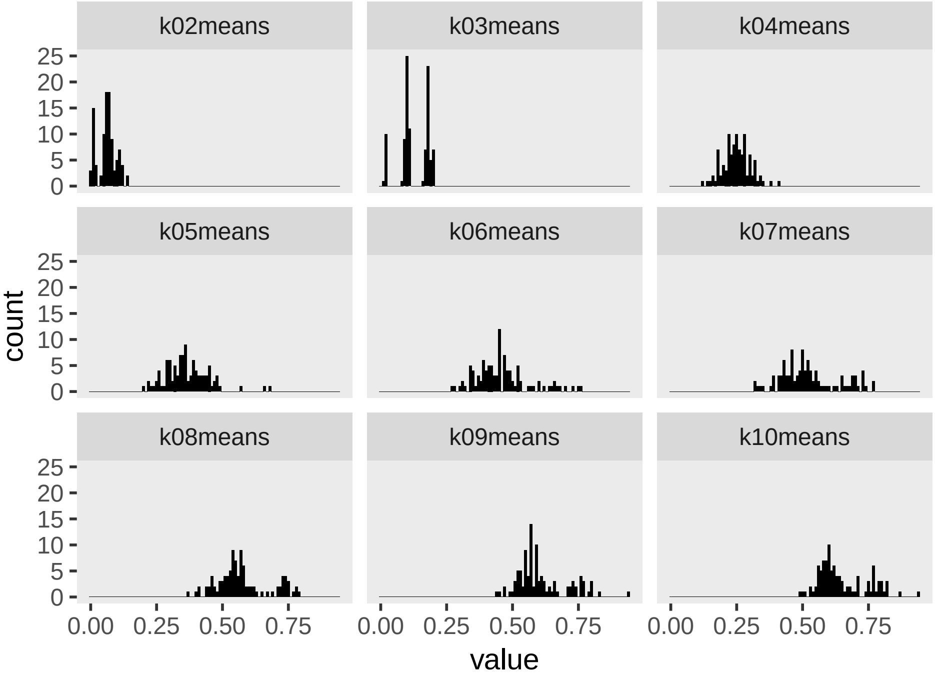

Figure 1: Histograms of the correlation similarity measure.
The distributions move closer to 1 as K increases.

Unfortunately, the histograms show that the algorithm does not function very well as intended when m is 0.2. This seems intuitive - if we're only sampling 20% of all data points, we cannot expect the correlation of random subsamples to be as high as if we were sampling a higher percentage that would naturally include more shared points. The data back this up. While increasing the k does move the distribution closer and closer to 1, these distributions never reach the high levels of correlation (> .9) that Ben-Hur witnessed with an m of 0.8. Moreover, the distributions never really tighten. There continues to be a large spread, with the only tight distributions appearing at k = 2 (likely due to the lack of k) and k = 3, which has a pretty clear separation into three areas.

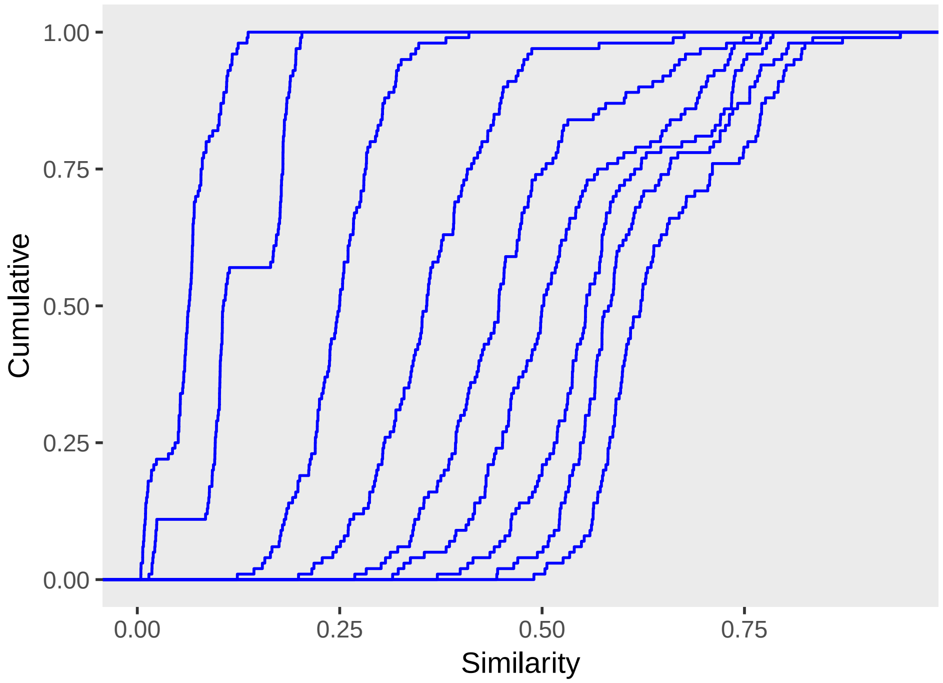

Figure 2: Overlay of cumulative distributions.
K increases from left to right in order from 2−10.

The cumulative distributions tell a similar story to the histograms. I decided not to include labels for each k because they appear in increasing order, with k = 2 having the lowest similarity and k = 10 having the highest. This is the opposite of most figures in the Ben-Hur paper, where instead the lowest ks are generally on the right and the highest ks are on the left. Additionally, figures in the paper have almost zero ks that have correlation measures lower than .5. In contrast, at m = .2, only a handful of ks have many values above .5, and very few of the ks have any correlation above .75 - only k = 9 and k = 10 have a substantive amount. Like the above histograms, we see here at k = 3 stands out. While the rest of cumulative distributions look like roughly the same distribution but with different ranges, the distribution for k = 3 has two clear steps, separating the distribution into three well-separated and easily identifiable areas.

3 Discussion

From this analysis, I would choose k = 3 as the number of clusters to use. I would choose three because of its tight distribution and clear separation of results, which shows a level of consistency that other ks do not express here. Regardless of my choice, however, I do not trust this method of stability at a low m. The original paper, by using a m of 0.8, shows ks with consistently high correlations that suggest computational stability. The above results show no such stability, however, at an m of 0.2. The correlations are low, and the distribution of each k is spread out. This makes using this method ineffective when using smaller subsamples.

Because the effectiveness of Ben-Hur's algorithm depends heavily on the size of your subsample, this means its utility is directly connected to your capacity. If you have the computing power to run something as computational rigorous as this algorithm, it can certainly be useful to helping you find a k that produces consistent results. If you do not have a lot of computing power, however, this method is effectively useless. If you don't have access to a computing cluster, processes like the Elbow Method or the Silhouette Method can help you arrive at a similar conclusion in a more practical manner. That being said, I do no doubt the stability of the findings from this algorithm with larger subsamples given the findings in the paper, and would certainly suggest using this algorithm if you have the ability because it provides you with evidence to make it easier to both choose and defend your choice of k.

Bibliography

Ben-Hur et. al., Asa. 2002. “A Stability Based Method for Discovering Structure in Clustered Data.” Pacific Symposium on Biocomputing, 6–17. http://www.cs.colostate.edu/~asa/papers/psb02.pdf.