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

SID: 3032130297

October 23, 2017

1 Introduction

In this lab, we use the data from the linguistic survey again, to test the stability of kmeans use the method outlined in Ben-Hur et al. [2001]. Section 2 including the reason to choose correlation has the similarity measure, and section 3 is about the efficiency of similarity algorithm. Section 4 contains the results and plots for stability of kmeans on the linguistic data, and section 5 is a little discussion about the method.

2 Similarity Measure

There are three similarity measures mentioned in the paper - correlation, Jaccard, matching. I tested them on some randomly simulated labeling vectors, and compared the results.

k	Correlation	Jaccard	Matching
3	0.3349611	0.2011732	0.556314
7	0.1485202	0.0802170	0.755174
10	0.1095358	0.0579396	0.819196

As shown in the table, when I increased the number of groups k, correlation and Jaccard went down together, but matching went up. Since all the data is generated randomly, the larger the k is, the more rambling the labels will be. Therefore, I will expect a smaller similarity score for larger k. The reason why the matching score is higher is it counts those "negative" matches, which means two points that are not in the same cluster in $L1$ also not in the same cluster in $L2$. And the probability of "negative" matches is really high when we have a large k. Moreover, since Jaccard similarity is a little bit too low, I will use correlation as the similarity measure in the rest of the report.

3 Runtime Comparison

I made 4 versions of correlation similarity function, three in R and one in C++. The first two versions store the matrix C. The first version uses two layers of 'for' loops (for n), and the second one only uses one layer of loop for k.

Version 3 (in R) and 4 (in C++) are the memory-effective versions without storing matrix C, they uses the same algorithm which runs in $O(k^2n)$ time. It uses the fact that the dot product (defined in the paper) of $L1$ and $L2$ is the sum of square of the number of points that belongs to cluster i in $L1$ and cluster j in $L2$ for any $i, j = 1, \ldots, n$.

I sampled 5000 points in the linguistic data, ran kmeans twice with different starting points, and tested the speed of similarity functions on the clustering results. My windows system cannot run 'Rcpp', therefore I compared the runtime on the server. The script I ran is 'runtime.R' and the result is in 'runtime.Rdata'. Here is the result.

```
## Unit: milliseconds
##                              expr          min           lq         mean
##   correlation_similarity(L1, L2) 70531.949496 70784.911627 72296.012000
##  correlation_similarity2(L1, L2)  1836.698288  2103.258565  2310.878821
##  correlation_similarity3(L1, L2)     2.874426     3.121754     5.561858
##            SimilarityCPP(L1, L2)     1.718924     1.760384     2.277264
##        median           uq          max neval
##  72050.703792 73213.730969 76238.474189    10
##   2294.285568  2463.744628  3114.473502    10
##      3.438740     4.987726    22.628555    10
##      1.776595     2.525224     4.971004    10
```

The average runtime of the original function is 72296 milliseconds, and the memory-efficient version in R cost 5.56 milliseconds. The C++ version is the fastest, the average runtime is 2.28 milliseconds, which is only a half of the corresponding R version and is nothing compared with the original one.

4 Stability for Linguistic Survey Data

Since in last lab, we found PCA is not that useful on this linguistic data, I performed kmeans directly on the dataset. I sampled 70% data ($m = 0.7$) each time, and calculate the similarity. I set the maximum number of clusters to consider $kmax = 10$ and the number of repeated iterations $N = 100$. The script runs on sever is 'parallel.R' and the result is 'stability.Rdata'.

From figure 1, we can see for $k = 2$, most of the correlations are between 0.5 and 0.6, while for $k = 3$, they are all concentrated around 0.34, and for larger k, the correlations are lower. In figure 2, we can see that the distance between $k = 2$ and $k = 3$ and the distance between $k = 3$ and $k = 4$ are quite large.

There are different reasons to support both $k = 2$ and $k = 3$. Personally, I will choose $k = 2$ for the following reasons. First, in the correlation for $k = 3$ is much lower than $k = 2$. Moreover in figure 2, the distance between $k = 2$ and $k = 3$ is larger than the one between $k = 3$ and $k = 4$.

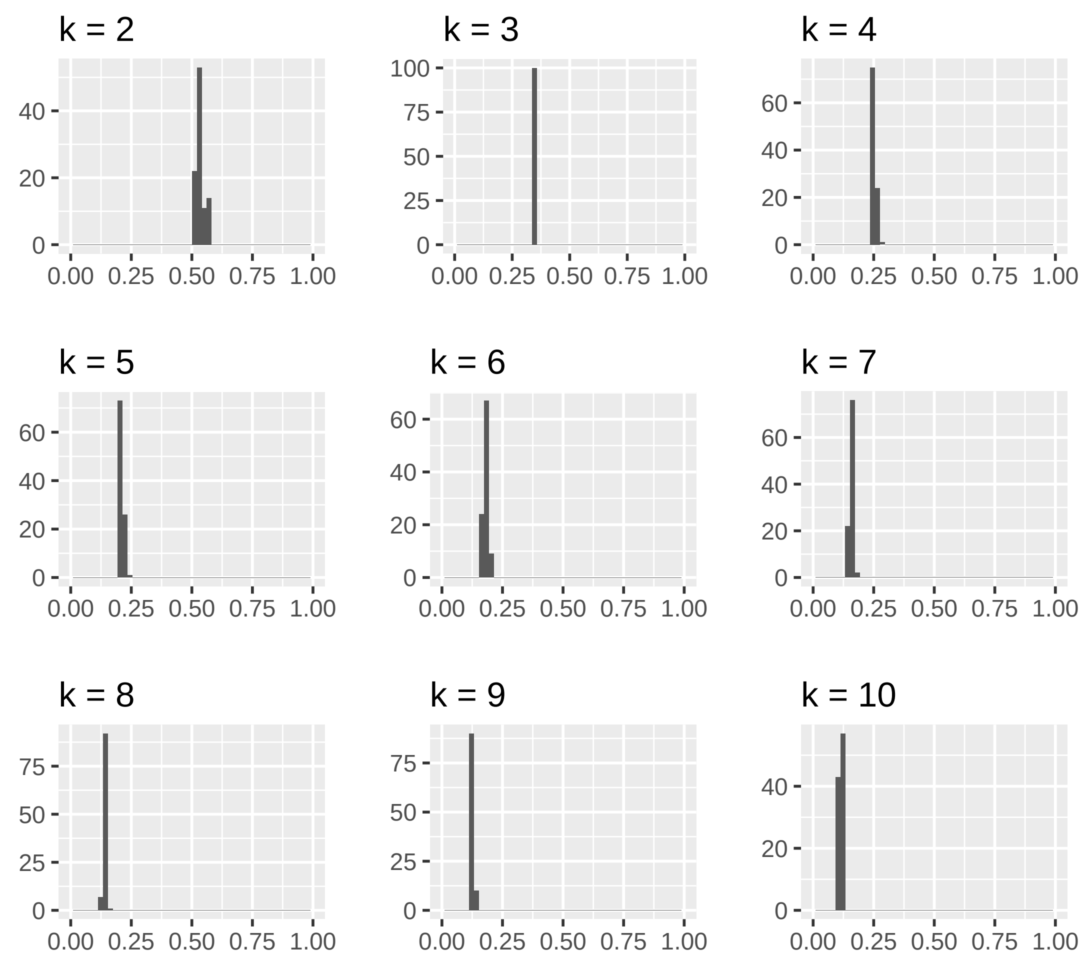

Figure 1: Histogram of the correlation similarity measure for different k.

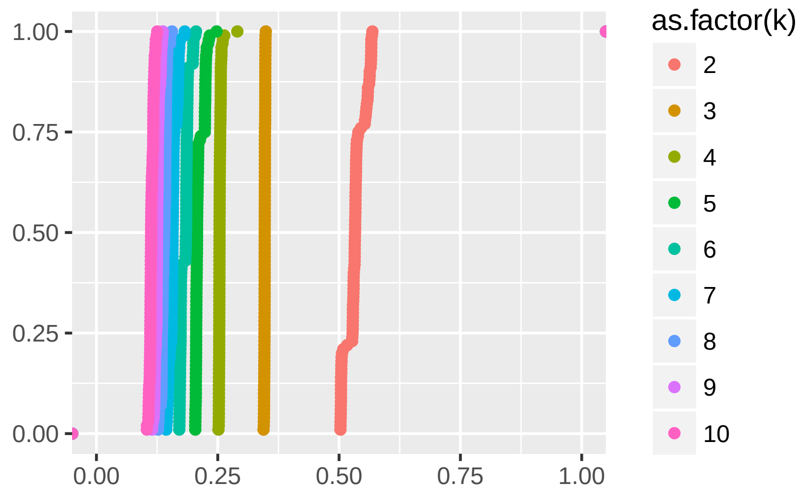

Figure 2: Overlay of the cumulative distributions for increasing values of k.

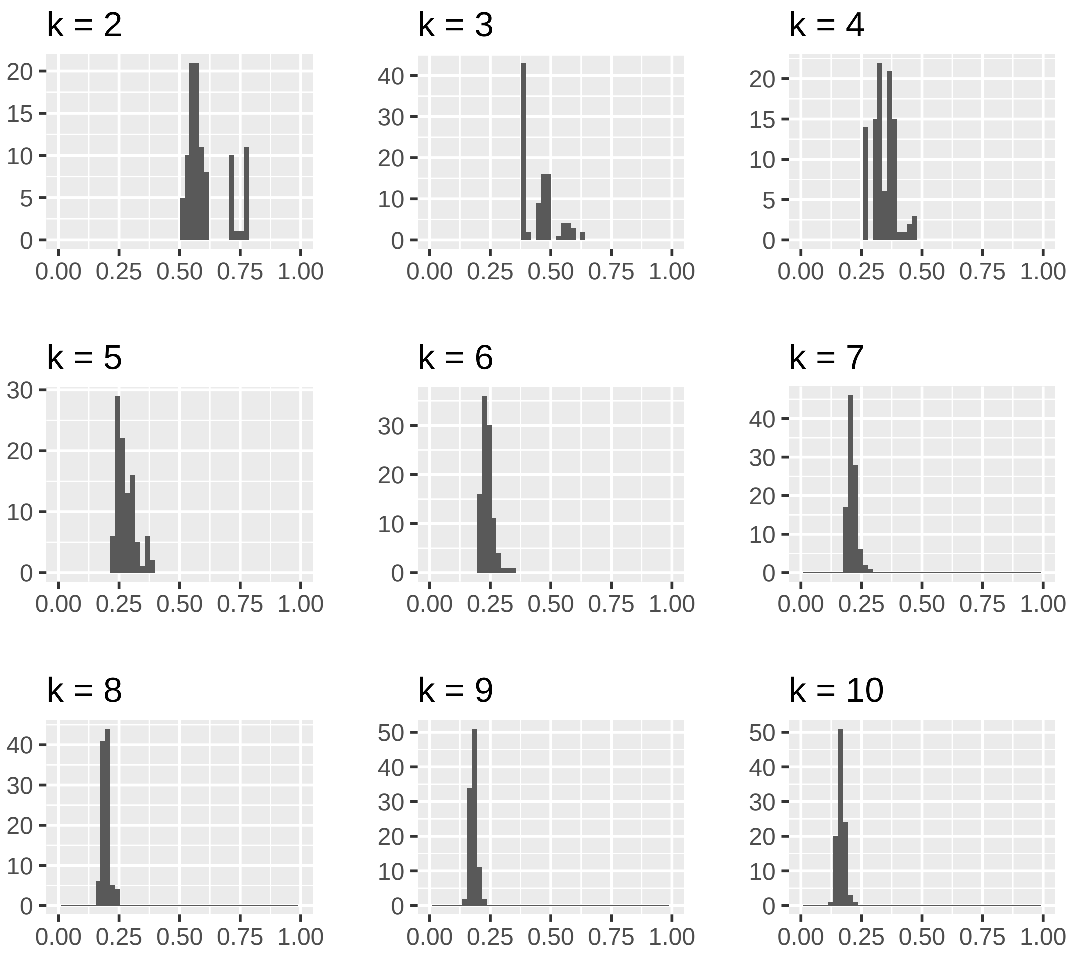

Figure 3: Histogram of the correlation similarity measure for different k on pca data.

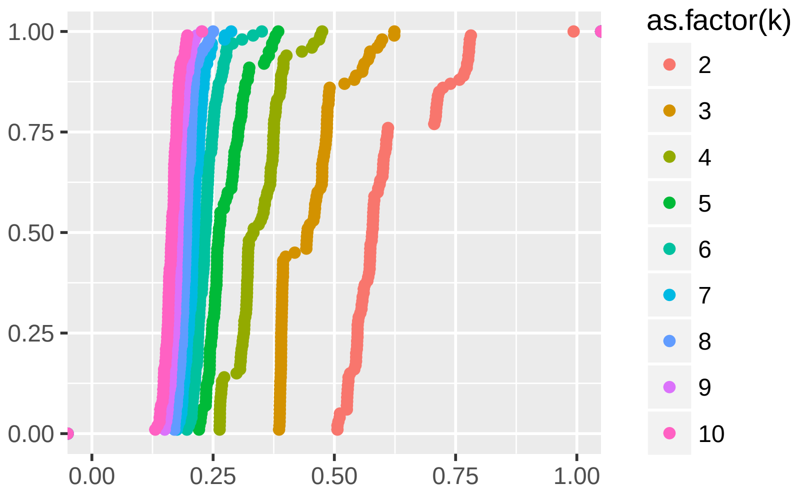

Figure 4: Overlay of the cumulative distributions for increasing values of k on pca data.

I also tried the same test on PCA data and used first 100 principle components.

The results are pretty much similar as on the original data, however, I may choose $k = 3$ based on these two figure. In figure 3, we can see that $k = 2$ has a wider spectrum of similarities than $k = 3$, and the correlations for $k = 3$ are not too low. In figure 4, the distance between $k = 2$ and $k = 3$ and the distance between $k = 3$ and $k = 4$ are very similar. Therefore, $k = 2, 3$ don't have many differences and I will choose the larger one since it gives more information.

5 Discussion

Generally, I trust this method. This is definitely a good practice to measure the stability of clustering. However, it may still have some issues there. For example, as I mentioned, the matching similarity goes to the opposite direction when I increase k, which may lead to a completely different decision when choosing k. Moreover, the choice of k should not completely decided by the stability. Larger k gives more information about the data, but usually has a lower similarity score and is less stable. There is a trade-off between these.

6 Conclusion

In this report, I used correlation as the similarity measure, and tested the stability measure procedure outlined in Ben-Hur et al. [2001] on the linguistic data. I was able to develop a memory-efficient algorithm for calculating the similarity, and implemented in both R and C++. I analyzed the stability and choice of k for linguistic data based on the plots I got, and discussed both pros and cons of the method.