Lab 3 Report: Parallel Computing

October 22, 2017

SID: 26968730

1 Introduction

In this report, we investigate the stability of k-means based on the work by Ben-Hur et al. [1], through which way to determine the optimal k. The method proposed by Ben-Hur et al. is implemented based on the binary-coded linguistic data. We parallelize the outer loop using *foreach*, and compute the similarity of clusterings for R version and C++ version separately. We will also compare the computing time of R and C++.

2 Parallel Computing

The goal of our work is to choose the optimal number of cluster. However, in each loop of k, it requires a fair amount of computation. To shrink the time and complete the computation with high efficiency, we parallelize the outer loop using *foreach*. From the paper by Ben-Hur et al. [1], the labelings $\mathcal{L}_i$ has the matrix representations $C^{(i)}$ and the inner product of two labelings is defined as

$$< \mathcal{L}_1, \mathcal{L}_2 >= \sum_{i,j} C^{(1)}_{i,j} C^{(2)}_{i,j}.$$

The cosine similarity measure is defined as

$$cor(\mathcal{L}_1, \mathcal{L}_2) = \frac{< \mathcal{L}_1, \mathcal{L}_2 >}{\sqrt{< \mathcal{L}_1, \mathcal{L}_1 >< \mathcal{L}_2, \mathcal{L}_2 >}}.$$

2.1 R version

To calculate the similarity between two membership vectors in R, we are able to reduce two double loops of dimension $q \times q$ to $k \times q$. This is a huge reduction of time since $k \ll q$. The method is that for the outer loop, we enumerate each cluster type $1 \leq i \leq k$, and then we find all the data points that belong to cluster i and mark these indexes. After that, we set the sub-matrix of the neighboring matrix with all the marked indexes 1. Through this procedure, we construct the neighboring matrix for each sub-samples. Then we add that two neighboring matrices and find the number of components which equal to 2 but minus the diagonal components.

2.2 C++ version

In C++ version, to avoid storing the huge $q \times q$ neighboring matrix, we use double loops with computation complexity $q \times q$. For the first sub-sample, we compare if the data points i and j are in the same cluster. If it is, then we look at the second sub-sample and if they are also in the same cluster, we add 1 to the final count result. Since C++ computes very fast, we would rather avoid storing the big neighboring matrix and increase the complexity.

2.3 Comparison between R and C++

When running R and C++ separately, C++ is about 500 times faster than R. Set $m = 0.3$ and $k = 3$, R version runs about 54.716 s per iteration while C++ version only needs 0.11 s per iteration.

3 Plots and best k

In this section, we do the parallel computing and plot the histogram of the correlation similarity measure. To have a clear picture of which k to choose, we also plot the cumulative distributions for different k.

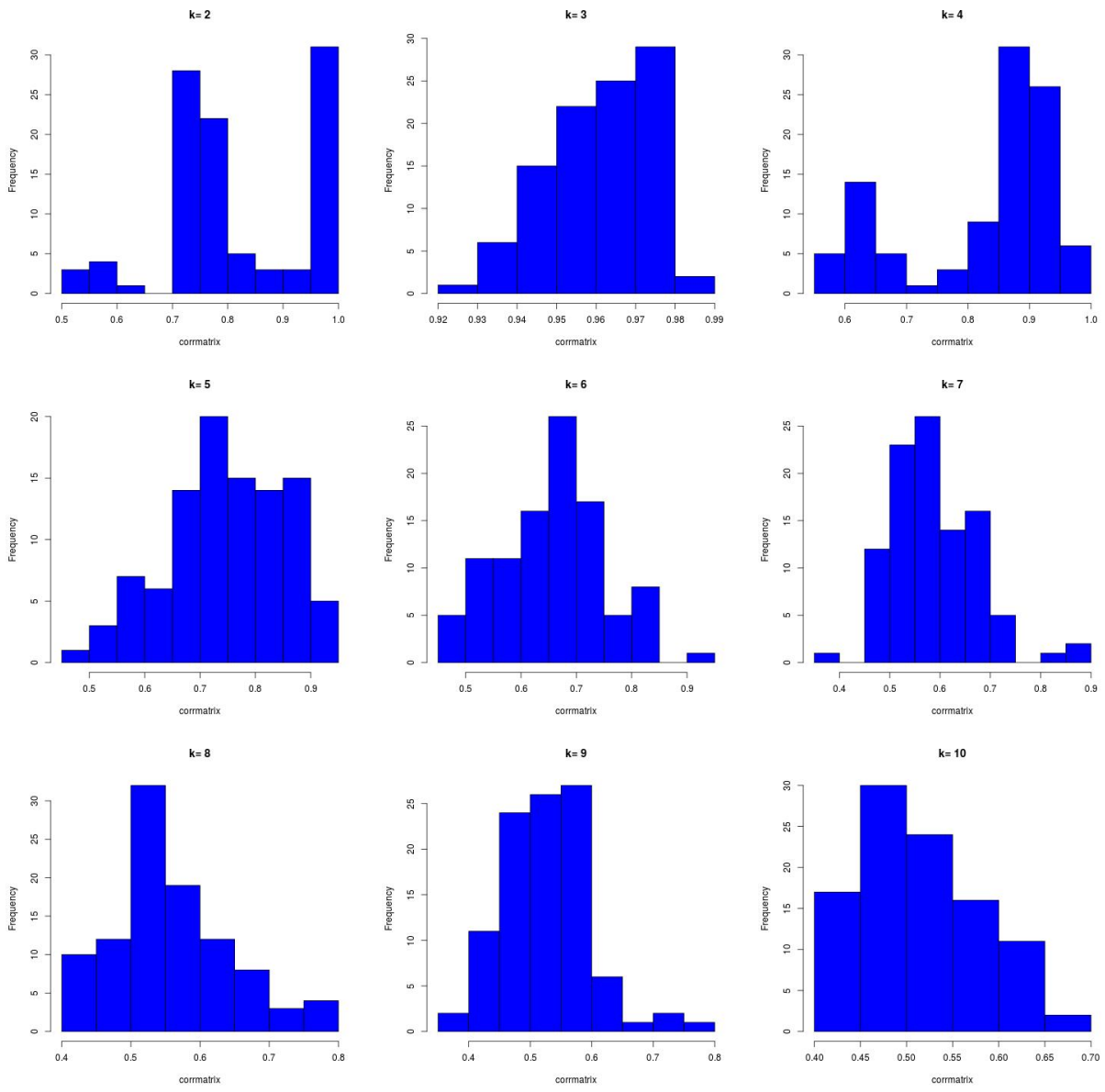

Figure 1: Histogram of the correlation similarity measure.

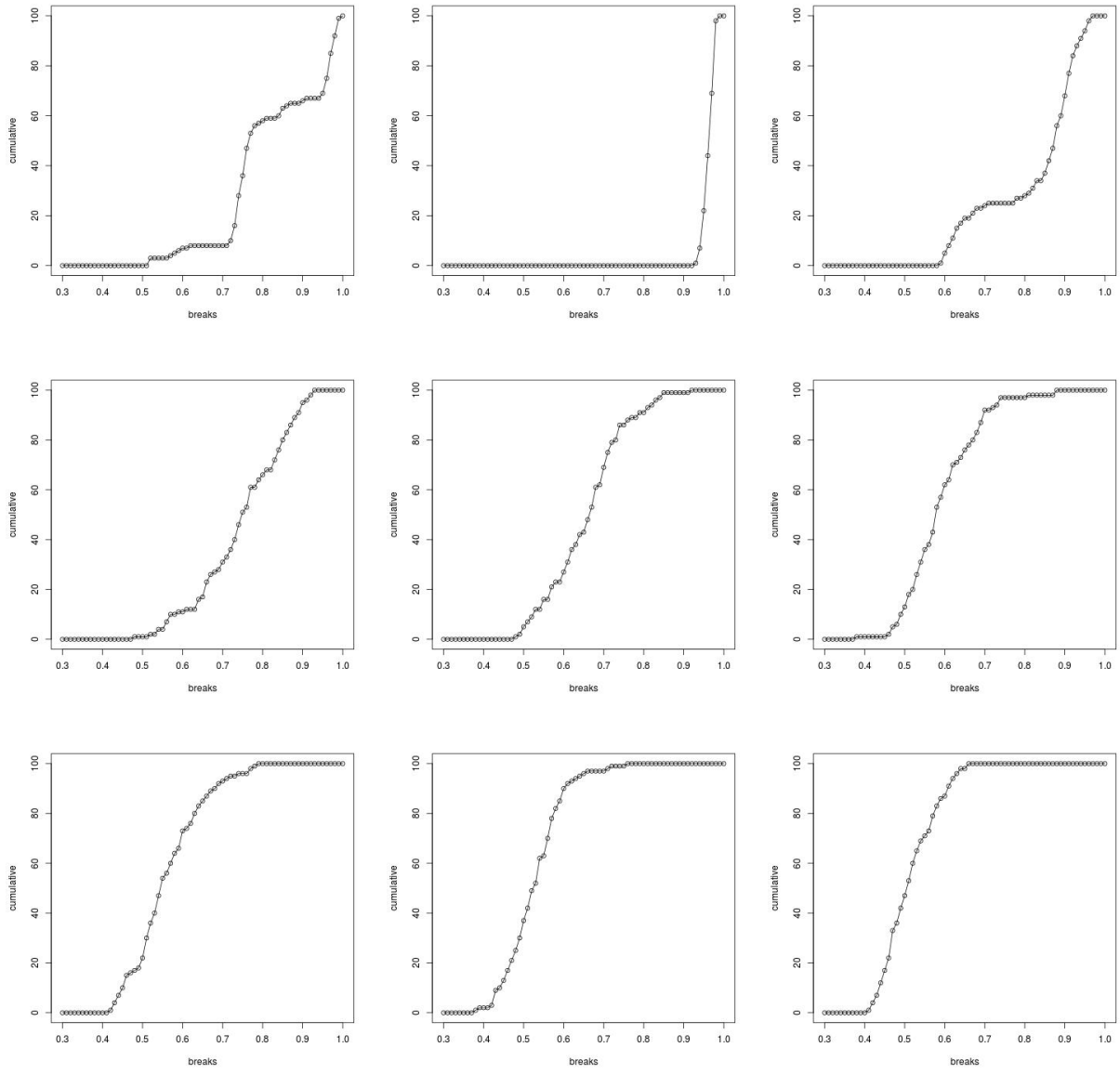

Figure 2: Overlay of the cumulative distributions for increasing values of k.

If we combine the above nine graphs, we get the following graph.

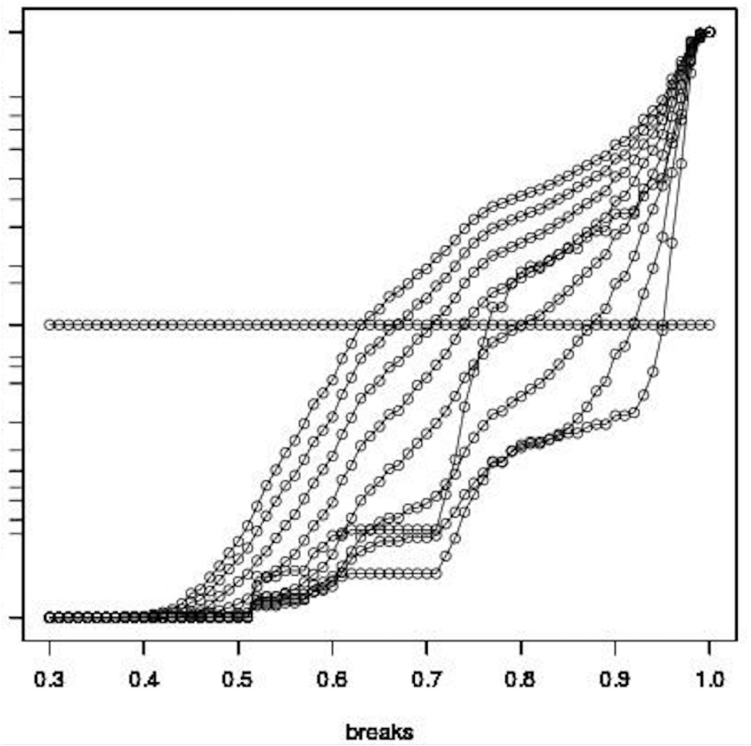

Figure 3: Overlay of the cumulative distributions for different k.

From graphs above, we find $k = 3$ is the optimal choice.

4 Discussion

In this lab work, we implemented the method proposed by Ben-Hur et al. to determine the optimal k in the kmeans clustering. Through parallel computing , we find $k = 3$ is the best choice. This result is consistent with the analysis in lab2. We can trust this method for some datasets but maybe not all datasets. For different choices of m, although within the range (0.2,0.8), the result may vary a lot.

References

[1] Asa Ben-Hur, Andre Elisseeff, and Isabelle Guyon. A stability based method for discovering structure in clustered data. In *Pacific symposium on biocomputing*, volume 7, pages 6–17, 2001.