STAT 215A – Lab 3

Kevin Benac

October 23, 2018

1 Introduction

In this lab, we investigate the stability of the k-means algorithm using a procedure outlined in Ben Hur et al. [2001] on the binary-coded linguistic data from Lab 2. This article presents similarity measures for comparing clusters obtained using different datasets as well as an algorithm to select the optimal number of clusters. Since this procedure is computationally intensive, we need to parallelize the code and run it on the UC Berkeley Statistics Computer Facility Cluster. We aim at identifying stable clusters.

2 Similarity Measures

In order to apply the model explorer algorithm described in Ben Hur et al. [2001] and pick an optimal k, we first need to come up with a measure of similarity between two clusters. We choose to use the Jaccard and Matching coefficients as given in Ben Hur et al. [2001] as they are intuitive and easy to implement.

We quickly recall here the definition of these similarity measures. Assume we have data $X = \{x_1, \dots, x_n\}$ where $x_i \in \mathbb{R}^d$, that we partitionned into k clusters: $\mathcal{L} = \{S_1, \dots, S_k\}$ (referred to as a *labelling* in Ben Hur et al. [2001]). Let $l_1, \dots, l_n \in \{1, \dots, k\}$ be the corresponding set of cluster labels for each observation. We can represent the clustering in a matrix C where

$$C_{ij} = \mathbb{1}_{\{\, l_i = l_j \text{ and } i \neq j\}}$$

For two clusterings $\mathcal{L}_1$ and $\mathcal{L}_2$ whose matrix representations are $C^{(1)}$ and $C^{(2)}$, then N_{ij} is defined for $i, j \in \{0, 1\}$ as the number of entries on which $C^{(1)} = i$ and $C^{(2)} = j$. The Matching coefficient is given as the proportion of entries on which the two clusterings agree, i.e.

$$M(\mathcal{L}_1, \mathcal{L}_2) = \frac{N_{00} + N_{11}}{N_{00} + N_{10} + N_{01} + N_{11}}$$

The Jaccard Coefficient is defined similarly except we ignore the N_{00}, that is

$$J(\mathcal{L}_1, \mathcal{L}_2) = \frac{N_{11}}{N_{10} + N_{01} + N_{11}}$$

The `similarity.R` file computes these coefficients in R and the `similarity.cpp` file does it in C++. In `R`, we have to vectorize the code. We shall note that $N_{00} + N_{01} + N_{10} + N_{11} = n^2$ so we only need to compute N_{00} and N_{11} in order to get the Matching similarity $M(\mathcal{L}_1, \mathcal{L}_2) = \frac{N_{00}+N_{11}}{n^2}$ and the Jaccard similarity $J(\mathcal{L}_1, \mathcal{L}_2) = \frac{N_{11}}{n^2 - N_{00}}$.

For the C++ version, we first note that we don't actually need to create and store the full C matrix because the matrix is symmetric and the diagonal is 0. Therefore, we only consider the upper triangular part then add up 2 instead of 1 when we check for N_{ij}. Since the diagonal is 0, we initialize $N_{00} = n$. We use of a nested for loop to compute N_{00} or N_{11} and hence, the Jaccard and Matching coefficients. We then do not have to store the $q \times q$ matrix.

`Jaccard R version:`

```
   user  system elapsed
  2.093   0.869   3.739
Jaccard C++ version
   user  system elapsed
  0.382   0.116   0.515
Matching R version
   user  system elapsed
  1.772   0.761   2.789
Matching C++ version
   user  system elapsed
  0.385   0.115   0.520
```

In order to test the efficiency of both codes, we take a subsample of 5000 entries in the `lingBinary` dataset and compute both the Jaccard and Matching coefficients using both the `R` and C++ versions. We find on average that the C++ version is 4-5 times faster than the `R` version. As a result, we prefer to use the C++ versions of the similarities when performing the parallelized stability tests to speed up the computational time.

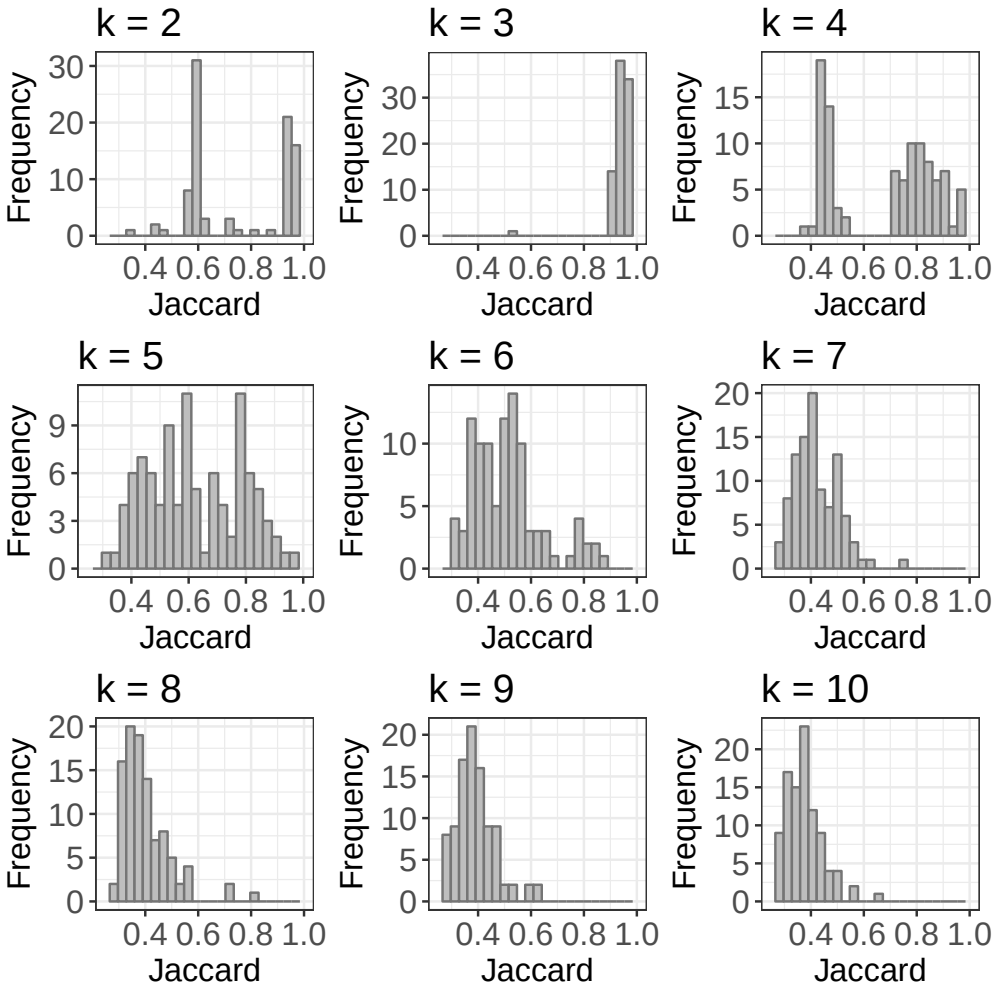

Figure 1: Histograms of Jaccard similarities for different values of k.

3 Stability of the Results

To assess the stability of the k-means algorithm, we use the Model explorer algorithm as described in Figure 2 of Ben Hur et al. [2001] We run k-means with k between 2 and 10 and repeat this 100 times each for two subsamples of the `lingBinary` dataset, then look at the clusterings of the intersection of the two subsamples and compute their similarity scores. This procedure was parallelized and ran on the computing cluster. In Figure 1, we look at the histograms for the Jaccard coefficient and in Figure 2, we look at the histograms for the Matching coefficient as in Figure 3 of Ben Hur et al. [2001]

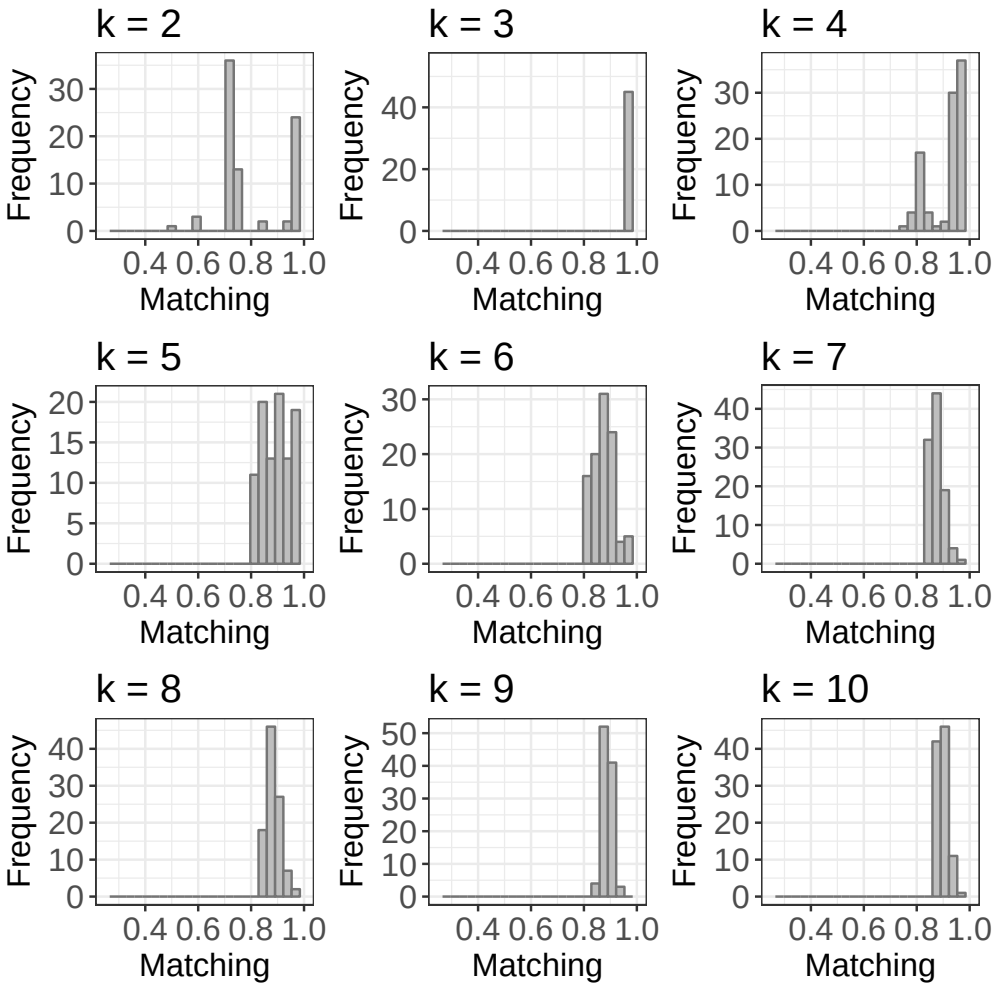

Figure 2: Histograms of Matching similarities for different values of k.

As explained in Ben Hur et al. [2001], the matching coefficient often varies over a smaller range than the Jaccard coefficient since the N_{00} term is usually a dominant factor. This is what we observe here. For every value of k, the Jaccard coefficient seems to be more spread out than the Matching coefficient. We also provide the empirical cumulative distribution function plots for both similarity measures as in Figure 3 of Ben Hur et al.

In Figures 3 and 4, we show the plots of the empirical CDF of the Jaccard and Matching Similarity measures color them by the number of clusters that were used. We are interested in identifying a number k of clusters and such that the k-means clusters are as tight as possible. First we look at Figures 1 and 2. An ideal number of clusters would be one with high similarity and low variance. This would mean that the cluster labels are consistent across the different runs. In both Figures 1 and 2, the value of k that best satisfies this criterion is 3, whether we consider the Jaccard or the Matching similarity measure.

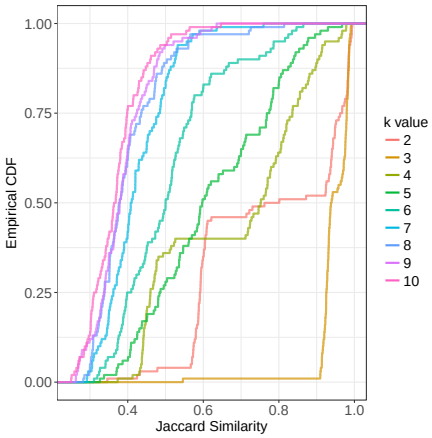

Figure 3: Empirical CDF of Jaccard similarity for different values of k.

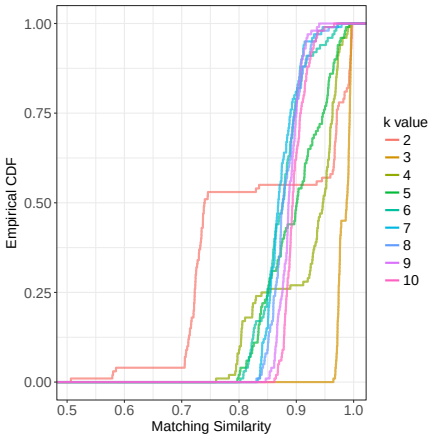

Figure 4: Empirical CDF of Matching similarity for different values of k.

Another thing we can look at is the distribution of similarity measures. An ideal number of clusters k would be one such that the empirical CDF's of the similarity measures is very close to 1. Again, it is clear according to this criterion that the optimal number of clusters to choose for good stability is $k = 3$. When looking in detail at the empirical CDF's, we see that for $k = 3$ clusters, the distribution starts to deviate from 0 at around 0.9 indicating good stability in the similarity measures. However one should note that although these two plots (histograms and empirical CDF's) are good for vizualization, it is unsurprising that the two are consistent with each other since both are based on the empirical distribution of the data so they carry pretty much the same information.

This is really consistent with what we found in the previous lab. We identified three major geographical groups. We also saw that $k = 4$ could work as well since in the previous lab, we found a cluster that did not relate to any geographical regions. Globally, I am not that surprised by these results and I trust them given what I found in the previous lab. An optimal number of clusters of $k = 3$ seems reasonable. The methodology is also very robust in that it can be applied to all sorts of clustering algorithms and the similarity measures are very intuitive as well. It would be very interesting to perform other clustering algorithms such as PAM to check whether the optimal choice of k is still 3 according to the Jaccard and Matching similarity measures.

4 Conclusion

We investigated the stability of the k-means using a procedure outlined in Ben Hur et al. [2001] on the binary-coded linguistic data from Lab 2. We were then able to pick an optinal number of clusters using the model explorer algorithm from Ben hur et al. [2001]. In order to run enough batches to assess the stability of k-means, we needed to write efficient code. We used C++ code to make it run faster and parallelized it on the UC Berkeley computing cluster. This enables us to pick $k = 3$ clusters which makes sense which the geographic clusters we found in the previous Lab.

5 Appendix: Documentation

5.1 Code

The code is presented in the `R` directory which contains 6 files.

- Two of them are from the similarity measures: `similarity.R` and `similarity.cpp`. Given clusters of two subsamples of the original data, they compute the C matrix as defined in Ben-Hur et al. [2001] and use it to compute the Jaccard and Matching coefficients. The `similarity.R` file corresponds to the R implementation and the `similarity.cpp` file to the C++ implementation which runs faster.

- The `stability.R` file uses the similarity file and parallelization in order to compute similarities for many runs on different numbers k of clusters for k-means given a specified number of runs, the number of cores used, the maximal k to try.
- The `main.R` file runs the stability function and writes a csv file where the results are stored.
- The shell scripts `Jaccard.sh` and `Matching.sh` are simply used to run the `main.R` file on the computing cluster. I used two nodes with 32 cpus. These two scripts just use the `R CMD BATCH -no-save '-args nCores=32 kMax=10 N=100 m=.8 similarityToReturn="jaccard"' main.R main.out` command to get the Jaccard coefficients and the other is the same up to changing the `similarityToReturn` argument to "matching" in order to obtain the Matching coefficients.

5.2 The Results

The results from the computing cluster are stored in two csv files (`Jaccard.csv` and `Matching.csv`) that are located in the `results` folder. They are presented in the form of two data frames with 100 rows corresponding to the different runs and 9 columns corresponding to the 9 values of k we tried.

6 References

[1] Asa Ben-Hur, AndrÃl' Elisseeff, and Isabelle Guyon. A stability based method for discovering structure in clustered data. In Pacific symposium on biocomputing, volume 7, pages 617, 2001.