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

SID: 3033031042

October 17, 2017

1 Introduction

Asa Ben-Hur et. al. describe a stability-based model for determining the optimal number of clusters, k, to use for clustering algorithms. Given some measure of similarity between two clusterings, the proposed method performs bootstrapping in order to generate two sub-samples of the original data set, and then compares the similarity between their resulting cluster assignments. Repetition of this process enables a distribution of similarity scores to be attained for each choice of k, which can then lead a scientist to determine which choice of k leads to the most "stable" clustering.

In this paper, we apply this notion of cluster stability on a linguistic survey data set that we previously had analyzed.

2 The model explorer algorithm

Asa Ben-Hur et.al. term their method of measuring cluster stability as the "model explorer algorithm". Here we will explain the components of the algorithm, and then explain how to parallelize its implementation.

2.1 Methods of similarity

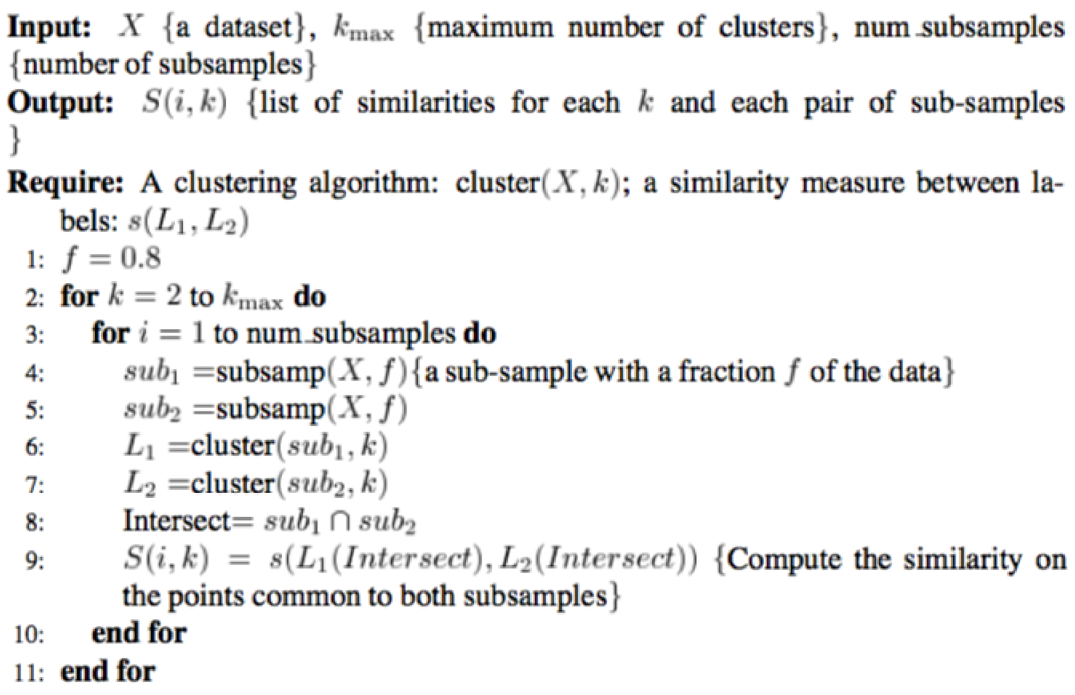
First, Ben-Hur et. al. define several methods of calculating similarity between two clustering results. Given the matrix representation C_{ij} where the $i-th$ row and $j-th$ column are a 1 if x_i and x_j belong to the same cluster and 0 otherwise, then we can define the following similarity measures between two different clusterings, $C^{(1)}$ and $C^{(2)}$:

- **Number of pairs of points clustered together**, or formally, $\sum_{i,j} C_{ij}^{(1)} C_{ij}^{(2)}$.
- **Correlation of co-pairings**, or formally, $\frac{\sum_{i,j} C_{ij}^{(1)} C_{ij}^{(2)}}{\sqrt{\sum_{i,i} C_{ii}^{(1)} C_{ii}^{(2)} \sum_{j,j} C_{jj}^{(1)} C_{jj}^{(2)}}}$
- **Matching coefficient**, or the fraction of entries on which the two matrices agree. Formally, $M(L_1, L_2) = \frac{N_{00}+N_{11}}{N_{00}+N_{01}+N_{10}+N_{11}}$ where N_{ij} is the number of entries on which $C^{(1)}$ and $C^{(2)}$ have values i and j respectively.
- **Jaccard coefficient**, or the fraction of entries on which the two matrices agree, excluding "negative" matches. Formally, this is $J(L_1, L_2) = \frac{N_{11}}{N_{01}+N_{10}+N_{11}}$.

The following analyses will use **the correlation of co-pairings** as the similarity measure between two clusterings.

2.2 Algorithm

The basic algorithm is provided in Figure 2 of the paper:



Input: X {a dataset}, $k_{\max}$ {maximum number of clusters}, num_subsamples {number of subsamples}
Output: $S(i,k)$ {list of similarities for each k and each pair of sub-samples }
Require: A clustering algorithm: cluster(X,k); a similarity measure between labels: $s(L_1, L_2)$

```
1:  $f = 0.8$
2:  for $k = 2$ to $k_{\max}$ do
3:      for $i = 1$ to num_subsamples do
4:          $sub_1$ =subsamp$(X, f)${a sub-sample with a fraction $f$ of the data}
5:          $sub_2$ =subsamp$(X, f)$
6:          $L_1$ =cluster$(sub_1, k)$
7:          $L_2$ =cluster$(sub_2, k)$
8:          Intersect= $sub_1 \cap sub_2$
9:          $S(i,k) = s(L_1(Intersect), L_2(Intersect))$ {Compute the similarity on
            the points common to both subsamples}
10:     end for
11: end for
```

Figure 2: The Model explorer algorithm.

The algorithm works by repeatedly drawing two subsamples from the original data set, and then calculating the similarity between pairs of points that were drawn in both subsamples

2.3 Similarity score implementation

To make the similarity score calculation as fast as possible, the entire similarity matrix C does not have to be calculated. A significant speed-up can be achieved if *only* pairs of observations that are clustered together in the first clustering are compared in the second clustering.

Here is a simple illustration:

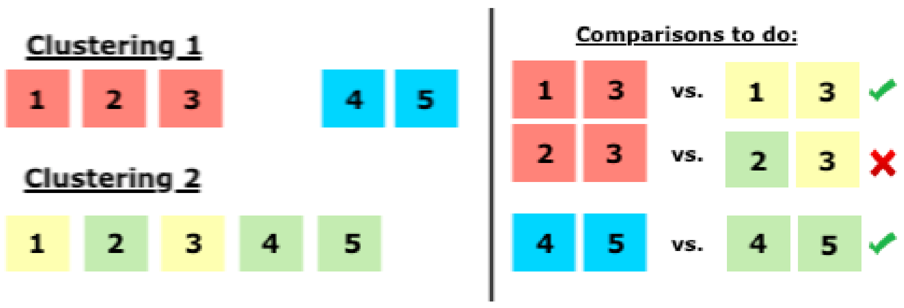


Figure 1: Visual illustration of the minimal comparisons necessary to calculate the correlation between two cluster assignments.

And then some pseudocode:

```
Data: cluster assignment 1, cluster assignment 2
Result: correlation similarity score
count = 0;
for ki in 1:k do
    current cluster = subset cluster assignment 1 to items in cluster ki;
    for i in 1:length(current cluster)-1 do
        for j in (i+1):length(current cluster) do
            if cluster assignment 2 for item i = cluster assignment 2 for
              item j then
                count += 1;
            end
        end
    end
end
return (normalized count);
```

2.4 Parallelization implementation

Since the above similarity score calculation involves nested for loops, using the "foreach" method in R to parallelize the outer loop in R was a natural choice.

2.5 R vs. C++ implementation

Using the same approach to calculate the correlation between two cluster assignments as shown in Section 2.3, we note that the C++ implementation is notably faster: the table below shows how much faster it was for three different proportions of data drawn.

proportion of data	R runtime / C++ runtime
0.2	31.9
0.5	30.8
0.8	32.2

This suggests that the runtime of the C++ version is roughly 30 times faster than the runtime of the R version. The agreement between the R and C++ numerical result was confirmed in each case.

3 Results

3.1 Optimal choice for k

The optimal choice for k is quite decisively 3, from figures 2 and 3.

Figure 2 highlights that the similarity scores between clusterings of different subsamples of data for $k = 3$ are very stable, as indicated by their concentration at 1. No other values of k comes close to this amount of stability.

Figure 3, whih shows the cumulative distributions of the correlation score for different values of k, also highlights that $k = 3$ has the most values that are concentrated near 1. Again, the CDF plot highlights that there is no other value of k that comes close to having similarity scores concentrated towards 1 as much as the orange line depicted by $k = 3$ does.

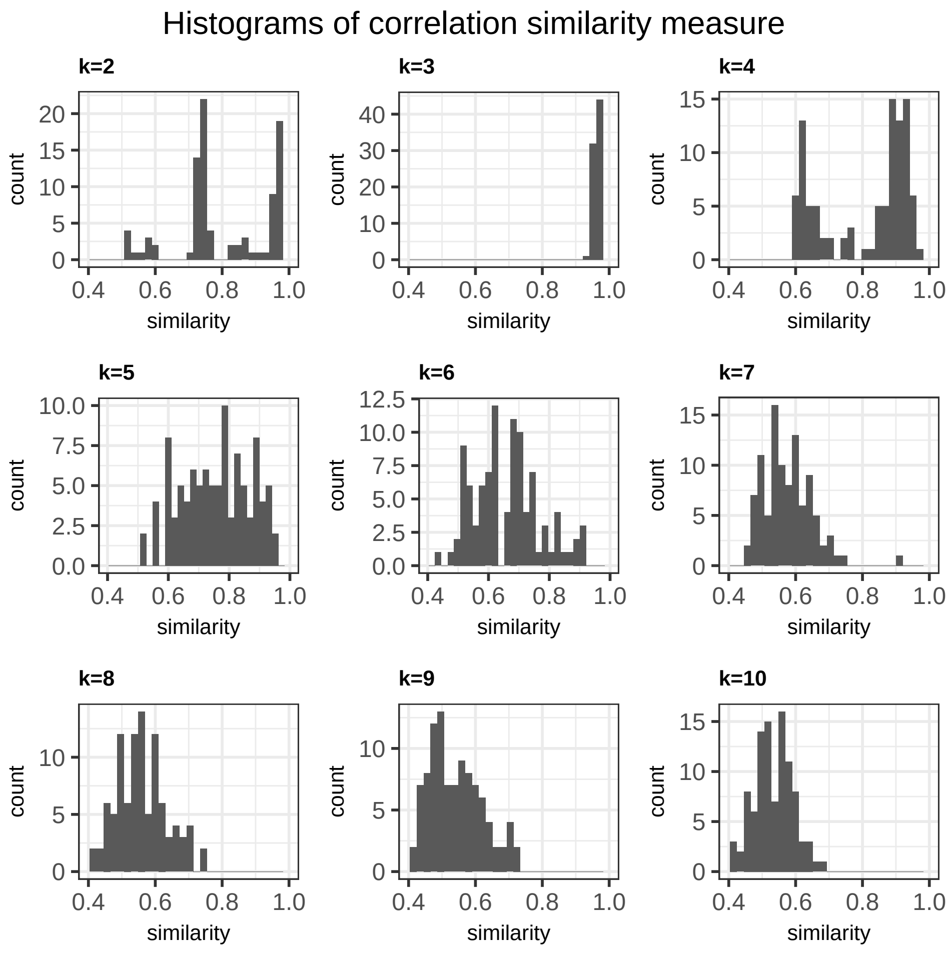


Figure 2: Histogram of the correlation similarity measure for different values of k

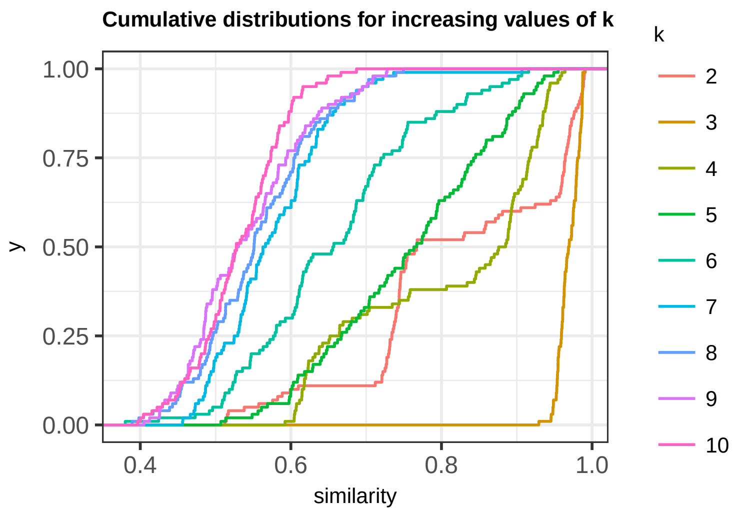


Figure 3: CDF of the correlation similarity measure for increasing k

3.2 Discuss whether you trust the method or not

Ben-Hur et. al. have developed a very strong method for choosing an optimal choice of k by assessing the stability of a clustering. Stability is important in that it signals that the algorithm will likely be robust to changes in the training data, and therefore that the method's predictions are generalizable to broader data sets.

In this particular data set, the optimal choice of $k = 3$ is very prominent. This may be surprising given that, in the previous lab, the optimal choice of k that produced the most "sensible" clusters was $k = 5$. However, in that case, the focus was on finding geographically-interpretable reasons and it may be understandable that $k = 3$ was too broad of a clustering to distinguish between the southeast and southwest, for instance.

This leads to a discussion of some of the algorithm's caveats:

- **Stability is not always the aim.** The primary reason to use this algorithm is to assess whether a clustering is stable, and for the stability to inform the choice of k. As discussed above, this may not always be the aim. If the aim is simply to learn more about what underlying structures might exist in the data, then a larger k that provides finer discrimination may be helpful, for instance.
- **Computational intensity.** In order for the two drawn subsets to have meaningful overlap, the percentage of the original population that is subsampled in each step must be large enough, but also small enough that computing the similarity score is not prohibitively expensive. For very large data sets, it may be hard to strike a balance between these two competing algorithmic needs. However, the parallelizability of the tasks can help, given sufficient computing resources.

4 References

1. Ben-Hur, Asa, Andre Elisseeff, and Isabelle Guyon. "A stability based method for discovering structure in clustered data." Pacific symposium on biocomputing. Vol. 7. 2001.