Hierarchical Clustering based on Voronoi edges: The R package HCV

Hsu Hao Yun : zoez5230100@gmail.com

Abstract

The R package `HCV` implements the hierarchical agglomerative clustering algorithm involving the spatial homogeneity by considering the Voronoi edges from R package `alphahull`. The Voronoi edges enforce the spatial proximity property at each step of iterations. Besides, we also offer several methods to find the optimal number of clusters under the consideration of spatial homogeneity.

1 Introduction

Today, analyzing spatial data is a crucial branch in data mining. Different from the conventional data structure, the spatial data are often analyzed with an assumption that if the two locations are close on geometry domain, then the features of these two locations are more strongly related than the distant pairs.

Clustering analysis is a practical means of dividing data into several subsets, with extracting the centroids or prototypes among the subsets. We than have a clear interpretation of the diversity of data. Two main branches of clustering method are partitional clustering and hierarchical clustering The former measures on the within-group sum of square (WSS) and require user specify the number of clusters in advance, while the letter builds the dissimilarity matrix and pulls together two clusters if the dissimilarity between these clusters is the minimum among all the pairs of the clusters formed in the last step. Nevertheless, the applicability of partitional clustering methods to spatial data is not clear. This package aims at proposing a novel method to find the clusters with spatial connectedness based on hierarchical clustering.

Another significant issue in cluster analysis is determining a proper number of clusters. The widely used methods are the internal indices and external indices. However, these traditional methods only depend on the within sum of squared matrix of non-spatial data attributes, which overlooks the condition on geometry domain. Hence, we propose a novel internal index call SMI (Spatial Mixture Index) for determining the optimal number of clusters, see Section 3.2.

2 Spatial clustering techniques

This section briefly reviews some literature on the spatial partition diagram and hierarchical clustering algorithm.

2.1 Voronoi diagram and Delaunay Triangulation

The Voronoi diagram is a efficient algorithm for constructing a set of partitions on geometry domain, see Figure 1. A best property of the diagram is that, when we add a new point A on the diagram, and we wish its nearest neighbor to be point B, we only have to locate the point A to the cell whose centroid is point B. Interestingly, if the point A locate on the boundary of two cells, then the nearest points of point A is the two centroids of these two cells.

The Delaunay Triangulation is a dual graph of Voronoi diagram, see Figure 2. All triangles in the graph are constructed by three nearest points, namely, whose edges represent the minimum cost path of the two endpoints.

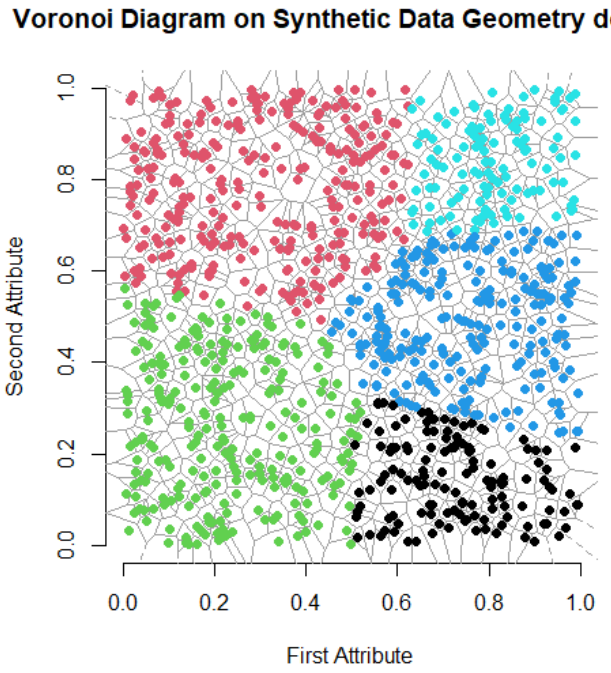


Figure 1: The Voronoi diagram on synthetic data

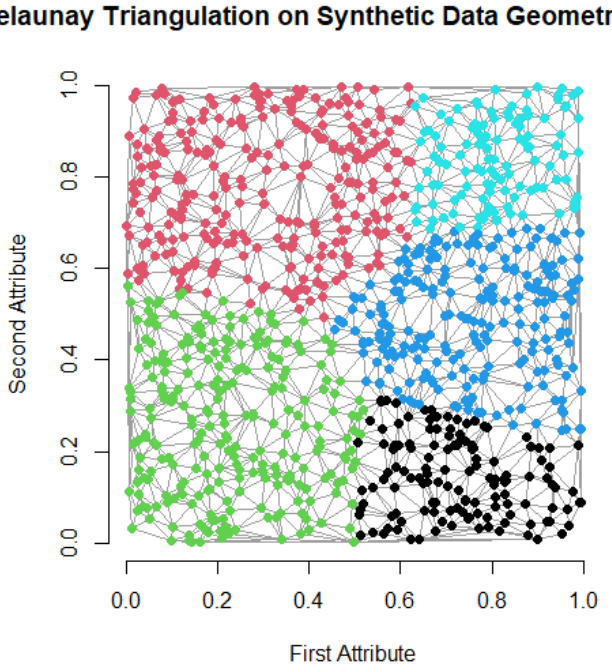


Figure 2: The Delaunay Triangulation on synthetic data

2.2 Hierarchical clustering

Hierarchical clustering algorithm is a stepwise grouping technique that iterates on the dissimilarity matrix and sequentially deals with two clusters exactly once in each step. Consequently, we have a

dendrogram at the end of the iteration. There are two commonly used hierarchical algorithms with opposite directions.

- AGNES : The algorithm works in a bottom-up manner. Initially, each individual is considered as a single cluster. Then it merges two clusters if they have the highest homogeneous.
- DIANA : The algorithm works in a top-down manner. Initially, the whole data is considered as a cluster. Then it split up into two clusters if they have the highest heterogeneous.

Let $x_i(s_i) = (x_{i1}, \ldots, x_{ip})'$ be a p-dimensional data located at s_i for $i = 1, \ldots, n$. Euclidean distance is often used to measure the distance or dissimilarity between pairs of points. A more general form of the measurement is Minkowski distance, as given in (1). Nevertheless, a cluster may contain more than one single point, and hence, several linkage methods for determining the between-cluster distance has been proposed. The well known methods, including single, complete, average, and Ward's method are all follow the Lance-Williams formula with different linear coefficient, see equation 2, where C_k is the cluster aggregated from cluster i and cluster j.

$$d(x(s_i), x(s_j)) = (\sum_{k=1}^{p} (|x_k(s_i) - x_k(s_j)|)^r)^{\frac{1}{r}} \tag{1}$$

$$d(C_h, C_k) = \alpha_i d(C_i, C_h) + \alpha_j d(C_j, C_h) + \beta d(C_i, C_j) + \gamma |d(C_h - C_i) - d(C_h, C_j)| \tag{2}$$

2.3 Hierarchical clustering based on Voronoi edges

The idea of our proposed HCV algorithm is generally based on the common edges of the Voronoi diagram. If two points occupy cells sharing a common edge in the Voronoi diagram, then they are considered to have spatial proximity and geometry connectedness. In each iteration, the major difference between HCV and a typical hierarchical agglomerative clustering algorithm is that, we only focus on geometry-connected groups, The next two candidate groups to be aggregated must share a common edge in Voronoi diagram, That is, a pair of clusters (h, k) can be merged in the t-th step only if $A_{hk}^{t1} = 1$, where the stepwise updated adjacency matrix A^t is recursively defined as

$$A_{ij}^{\{0\}} = A_{ij} \begin{cases} 0, & \text{if cluster } i \text{ and cluster } j \text{ have no common Voronoi edge} \\ 1, & \text{if cluster } i \text{ and cluster } j \text{ have an common Voronoi edge} \end{cases} \tag{3}$$

$$A_{hk}^{\{t\}} = \begin{cases} 0, & \text{if } A_{jk}^{\{t-1\}} = 0 \text{ and } A_{ik}^{\{t-1\}} = 0 \\ 1, & \text{if } A_{jk}^{\{t-1\}} = 1 \text{ or } A_{ik}^{\{t-1\}} = 1 \end{cases} \tag{4}$$

The general steps for HCV algorithm is as follows.

1. Construct the Voronoi diagram on geometry domain
2. Build the dissimilarity matrix D
3. Build the adjacency matrix A according to (3)
4. Aggregate i group and j group if $\arg \min_{i,j,A_{ij}=1} D_{ij}$
5. Update the dissimilarity matrix and the adjacency matrix according to (2) and (4)
6. Repeat step 3 to 5 until every points are in one single cluster, or no clusters share any common edge with each other

3 Implementation

There are two main functions in the `HCV` package, `HierarchicalVoronoi` and `SMI`. The former is used to implement the HCV algorithm, while the letter is used to determining the optimal number of clusters under the constraint of spatial connectedness.

3.1 HierarchicalVoronoi

For implementing the HCV algorithm, we have to specify the constraint domain (geometry domain) which is used to construct the Voronoi diagram and the feature domain (non-geometry domain) which is used to construct the dissimilarity matrix. In the function `HierarchicalVoronoi`, the data type of constraint domain and feature domain must be matrix objects: the constraint domain is a $n \times 2$ matrix with n be the sample size and the feature domain is a $n \times p$ matrix with p be the number of features, It the user already has the dissimilarity matrix or adjacency matrix, (s)he may set `diss = 'precomputed'` for input optimization domain as an $n \times n$ matrix and `adjacency = True` for inputting feature domain dissimilarity as an $n \times n$ adjacency matrix. Finally, choose a linkage method for the distance between clusters, the default is `'ward.D'` method. The output of HCV algorithm is a list with class `hclust`; therefore, the R base function `cutree` can be used to obtain the cluster labels of each point. The following example codes demonstrate an easy way to use the HCV algorithm.

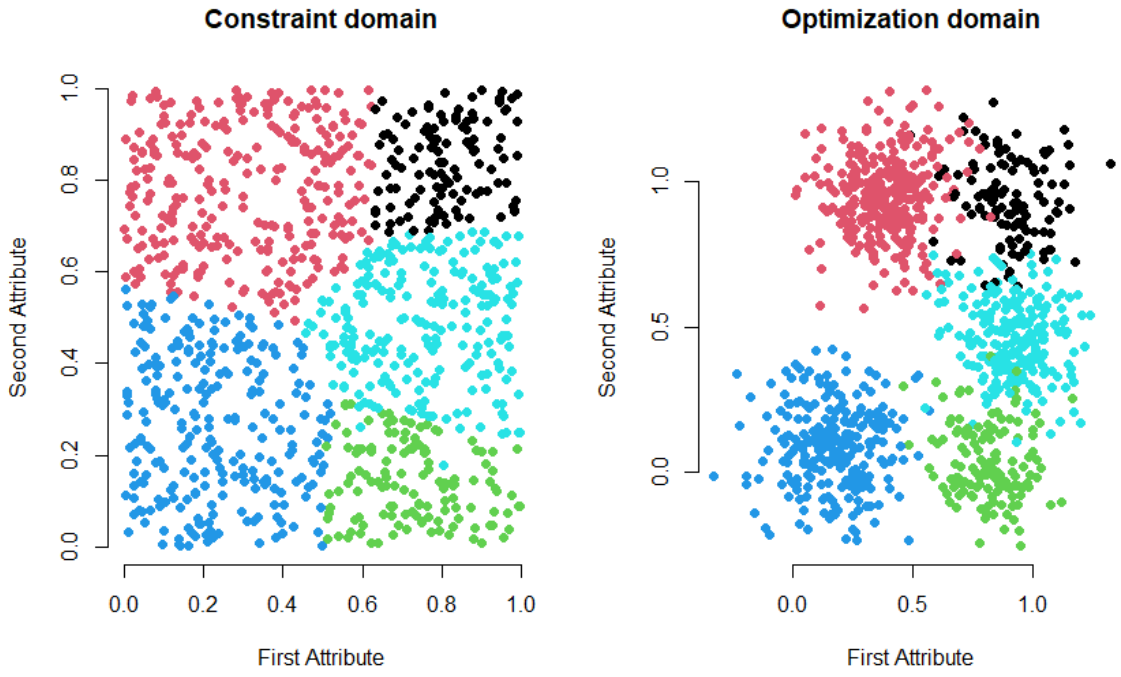
```
> # Generate 1000 points with constraint domain (geo) and optimization domain (feat)
> spatialData <- synthetic_data(5,100,0.02,1000,2,2)
> result <- HierarchicalVoronoi(spatialData$geo,
+                               spatialData$feat, linkage = 'ward.D')
> result
Cluster method   : ward.D
Distance         : euclidean
Number of objects: 1000

> names(result)
[1] "label_matrix" "merge"        "height"       "method"       "dist.method"
[6] "labels"       "order"

> class(result)
[1] "hclust"

> par(mfrow = c(1,2))
> plot(spatialData$geo, col = cutree(result, 5), pch = 19, main = 'Constraint domain',
+      frame = F, xlab = 'First Attribute', ylab = 'Second Attribute')
> plot(spatialData$feat, col = cutree(result, 5), pch = 19, main = 'Optimization domain',
+      frame = F, xlab = 'First Attribute', ylab = 'Second Attribute')
```

The `label_matrix` is the matrix which stores the result of `cutree`, and `result$label_matrix[i, ]` is equivalent to `cutree(result, i)`. The other attributes is exactly the same to the attribute of class `hclust`.



3.2 SMI

SMI is an internal index involving the property of spatial proximity by considering the edge length of Delaunay Triangulation. Let $x_i(s_i) = (x_{i1}, \ldots, x_{ip})'$ be a p-dimensional data located at s_i for $i = 1, \ldots, n$, SMI is given in (5), where μ_k and δ_k represent the centroids of the k-th cluster for the feature domain and the constraint domain respectively, and e_k is the average Delaunay edges length in the k-th cluster.

$$\mathrm{SMI} = \frac{1}{K}\sum_{k=1}^{K} f(\alpha_k)\mathrm{WSS}^k_{feat} + (1 - f(\alpha_k))\mathrm{WSS}^k_{geo} \tag{5}$$

$$\mathrm{WSS}^k_{feat} = \sum_{x_i \in C_k} (x_i - \mu_k)'(x_i - \mu_k) \tag{6}$$

$$\mathrm{WSS}^k_{geo} = \sum_{x_i \in C_k} (s_i - \delta_k)'(s_i - \delta_k) \tag{7}$$

The definition of α_k and function f are as follows.

$$\begin{aligned} \mathrm{WSS}^k_f &= \sum_{x_i \in C_k} (x_i - \mu_k)'(x_i - \mu_k) \\ \mathrm{WSS}_f &= \sum_{n=1}^{K} \mathrm{WSS}^n_f \\ \alpha_k &= \frac{K e_k - \sum_{j=1}^{K} e_j}{\sum_{j=1}^{K} e_j} \\ f(x) &= (1 + e^{-x})^{-1} \end{aligned} \tag{8}$$

A clearly explaination of the formula will be posted on my github soon, and the usage example code is shown below :

```
> SMI(spatialData$geo, spatialData$feat, result, 30)
$bestCluster
```

```
[1] 3

$index
 [1] 638.606660 231.535392 119.637256  74.297295  59.297407  48.714523  40.508281
 [8]  34.691950  29.736230  26.683105  24.074248  21.772144  19.968204  18.481191
[15]  16.810873  15.682021  14.289012  12.247916  11.508806  10.445373   9.909160
[22]   9.341543   8.862649   8.407029   7.911264   7.542482   7.220705   6.801251
[29]   6.478649

$ratio
 [1] 0.63743662 0.48328739 0.37897861 0.20189009 0.17847129 0.16845576 0.14358374
 [8] 0.14284928 0.10267358 0.09777189 0.09562514 0.08285540 0.07446905 0.09037935
[15] 0.06715011 0.08882842 0.14284375 0.06034577 0.09240166 0.05133500 0.05728208
[22] 0.05126495 0.05140901 0.05897029 0.04661480 0.04266201 0.05809034 0.04743286
```

The `SMI` function outputs the value of `SMI` ratio from cluster 2 to the maximum cluster you specify (default is `maxk`=30), in which ratio is the vector of ratios for different K, and the best cluster is determined by selecting the maximum ratio cluster.