FunctionDescription of AdjClustBand_heap

Parismita Das

25 March 2017

General Description for adjClustBand_heap Implementation

The adjClustBand_heap function takes in the Cosine Similarity Matrix, diagonal elements along with its width and no of rows as input.

As constrained Hierarchical Clustering has a time complexity of O(n^2) which is very costly. It taked 22 minutes to calculate 32768 SNPs. Furthermore, it takes 4.5 hours (on a standard 2.2 Ghz single CPU) to analyze a whole genome of 500k simulated SNPs (for Affymetrix 500k arrays) genotyped on 100 individuals.

Hence the quadratic complexity of the adjacency-constrained hierarchical clustering remain unsatisfactory for many reasons.

To aviod this situation We introduce of a parameter h aims to control the maximum lag between items X.i and X.j for similarity calculations. Thus, the similarity measures are computed between these two items only if i and j differ by no more than h. To implement this we use a pencil trick which calculates the distance in complexity of O(1)

Furthermore we use MinHeap for finding the best fusion. Thus reduce the complexity from O(p^2) to O(plogp).

In the adjClustBandheap.R file, the input is converted into a left and right matrix of p*h size. And Heap of p-1 size is build. The output is a hclust class object, Which givw similar results as rioja chclust(coniss linkage) and with better optimisation.

The Algorithm Used

Algorithm 8 The **optimized** algorithm of the Ward's constrained hierarchical clustering applied to a h-band similarity matrix

procedure cWARD($\mathbf{X} \in \mathbb{R}^{n \times p}$, Sim, h)
  Calculate the two $p \times h$ arrays of pencils sums ▷ $\mathcal{O}(ph)$
  Initialize the chained array Tab
  $heap \leftarrow$ `buildHeap`$(1 : (p-1), D)$ ▷ $\mathcal{O}(p \log(p))$
  $jj \leftarrow p$
  for $step = 1$ **to** $p - 1$ **do**
    while $(!Tab[valid, heap[1]])$ **do**
      $heap \leftarrow$ `deleteMin`$(heap)$ ▷ $\mathcal{O}(\log(p))$
    end while
    $posMin \leftarrow heap[1]$
    $i^\star \leftarrow Tab[Cl1, posMin]$
    $heap \leftarrow$ `deleteMin`$(heap)$ ▷ $\mathcal{O}(\log(p))$
    $d_1 \leftarrow D(C_{i^\star-1}, C_{i^\star} \cup C_{i^\star+1})$ ▷ $\mathcal{O}(1)$
    $d_2 \leftarrow D(C_{i^\star} \cup C_{i^\star+1}, C_{i^\star+2})$
    Add the distances d_1 and d_2 to Tab
    $heap \leftarrow$ `insertHeap`$(heap, jj, D)$ ▷ $\mathcal{O}(\log(p))$
    $heap \leftarrow$ `insertHeap`$(heap, jj + 1, D)$ ▷ $\mathcal{O}(\log(p))$
    Update the neighbors of $C_{i^\star-1}$ and $C_{i^\star+2}$ in Tab
    Set $Tab[valid, posMin]$, $Tab[valid, posL]$ and $Tab[valid, posR]$ to `FALSE`
    $jj \leftarrow jj + 2$
  end for

Function Description

- percDown : Rebalancing of elements after insertion, so that the Min-Heap property is maintained.
- deleteMin_C : Delete the minimum position after merging of the clusters.
- insertHeap_C : Insert the cluster after obtaining the minimum distance from `distance_C`
- neighborCl_C : Give the adjacent position of cluster.
- neiNeighborPos_C : Give the neighpour position of adjacent position of cluster.
- pencil_C : Calculate the pencil shaped area by using partial sum of similarity, used for distance calculation.
- distance_C : Calculate distance using Ward's criteria using Lance-William method and pencil implementation.
- cWardHeaps : The actual function that clusters according to the given similarity matrix using Min-heap for merging of clusters and ward's criteria for distance calculation required for merging of clusters.
- .toMatL : creates p*h matrix to calculate partial sum of similarity

- .toMatR : creates p*h matrix to calculate partial sum of similarity

The output has attributes such as traceW, gain, merge, height, label, order which is calculated in cWardHeaps function. That are used to create a 'hclust' object in the R interface of the implementation.

The Functions Dependency Graph