Efficient Hierarchical Clustering for Palsar Image Analysis

Author: Beaulieu Jean-Marie

Conference: Advanced SAR Workshop 2019, Canadian Space Agency
Montreal (Saint-Hubert)
1-3 Oct. 2019


© 2019 Jean-Marie Beaulieu
1. Hierarchical clustering

The agglomerative hierarchical clustering algorithm starts by assigning each data point to a distinct cluster. For \( N \) data points, we initially have \( N \) clusters. Then, iteratively, the number of clusters is reduced by 1 by merging the 2 most similar clusters. At the end, there is 1 cluster containing all the data points. At each iteration, we consider all pairs of clusters \((C_i, C_j)\), calculate a similarity measure or distance for each pair \( (D_{i,j} = D(C_i,C_j)) \) and merge the 2 clusters which are the most similar or have the smallest distance.

The algorithm is general but the distance \( D \) should be correctly defined for each application. \( D \) is a between cluster distance with varying cluster sizes. It could be related to inter-cluster distances \( d(x,y) \). The distance between cluster centers (mean values) is often used, \( D(C_i,C_j) = d(m_i,m_j) \) where \( m_i \) is the mean value of the \( C_i \) cluster data points.

Iterative clustering techniques start from an initial partition and iteratively improve it. The often used K-means algorithm starts with \( K \) center positions. Each data point is then assigned to the closest center. For each center, a new position is calculated from the mean values of points assigned to the center.

2. Computing time

The computing time of the K-means is of the order of \( K \times L \times N \), where \( K \) is the number of centers, \( L \) the number of iterations and \( N \) the number of pixels. The algorithm is fast because \( K \) and \( L \) are small (\( \leq 10 \)) even if \( N \) is large. \( N = 10^6 \) for a 1000\( \times \)1000 image. The computing time increases linearly with \( N \).

The hierarchical clustering calculates a \( N \times N \) distance matrix at each iteration, where \( N \) is the number of clusters. The computing time is of the order of \( \sum_{n=N}^{2} n^2 = \alpha N^3 \). We can expect a computing time of the order of \( N^2 \) because only the line and column associated with the last merge need to be updated. For large \( N \), the distance matrix will be too large to be stored (10^{12} = 1 \text{ Tera}).

We also need to find the minimum value of the distance matrix to identify the next cluster pair to merge. To make it \( \log_2 N \) instead of \( N^2 \), with each cluster, we store the label of its best neighbor. Then, we use a head sort tree to find the minimum among the best neighbors. Moreover, there is no need to keep the distance matrix.

3. Polsar criterion

The log likelihood ration is used to calculate the distance between the covariance or coherency matrices of two cluster mean values.

\[
D_{i,j} = 2 \ln \left( \frac{\Sigma_i + \Sigma_j}{2} \right) - \ln |\Sigma_i| - \ln |\Sigma_j|
\]

To reduce the computing time, a preselection function could be used at initialization and merging.

4. Neighbor lists and grid

For the merging steps, we can use only the \( k \) best neighbors for each cluster. The merging time is then reduced to \( kN \). Most of the time will then be used to initialize these lists before the merging. A grid is used to reduce the initialization. A subset of the features is selected (5D) and divided into cells. For a cluster, only the neighbor cells are used to build its list.

The following table shows the computing time for different image sizes. Lists of 25 neighbors and a 5D grid are used. It is useful to use both the lists for the merging steps and the grid for the initialization of the lists. The computing time reduction is important. The hierarchical segmentation is faster, the average number of neighbors is much smaller (\( \approx 8 \)).

<table>
<thead>
<tr>
<th>Image Size</th>
<th>segmentation</th>
<th>grid+list</th>
<th>selection+list</th>
<th>selection</th>
<th>list</th>
<th>no list</th>
</tr>
</thead>
<tbody>
<tr>
<td>300x400</td>
<td>0s 400ms</td>
<td>15s 200ms</td>
<td>2min 50s</td>
<td>8min 21s</td>
<td>18min 36s</td>
<td>42min 20s</td>
</tr>
<tr>
<td>600x400</td>
<td>0s 830ms</td>
<td>31s 500ms</td>
<td>11min 31s</td>
<td>35min 18s</td>
<td>74min 42s</td>
<td>177min 29s</td>
</tr>
<tr>
<td>600x800</td>
<td>1s 780ms</td>
<td>1min 7s 400ms</td>
<td>35min 3s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000x1000</td>
<td>3s 860ms</td>
<td>2min 31s 300ms</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>