SAR Image Segmentation using Spectral Clustering

Name: Giacomo Fratus

Abstract:

Aerial synthetic aperture radar (SAR) images are interesting because they produce high contrast images in varying weather conditions. Segmentation of these images can be done in several ways, but for this report, spectral clustering and k-means will be compared. K-means clustering and spectral clustering have similar objectives, to cluster data into meaningful groups. K-means tends to cluster points that are close together, thus favoring data with spherical clusters. Spectral clustering works by partitioning a connected graph, and thus favors clustering points that are highly connected in the graph. The benefits and drawbacks of each method will be evaluated on several aerial images.

Method:

Each image was transformed into a data vector, containing the pixels ordered column-wise.

$$Image = \begin{bmatrix} p_1 & p_5 & p_9 & p_{13} \\ p_2 & p_6 & p_{10} & p_{14} \\ p_3 & p_7 & p_{11} & p_{15} \\ p_4 & p_8 & p_{12} & p_{16} \end{bmatrix} \Rightarrow \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ \vdots \\ \vdots \\ \vdots \\ p_{14} \\ p_{15} \\ p_{16} \end{bmatrix} = Data\ vector$$

P_i represents the i-th pixel intensity. Thus, an image of size mxn will result in a data vector of size mnx1
Let mn = z

K-means

Performing K-means on this data is simple, and involves clustering the pixels into k groups based on their intensity. Pixels of similar intensity will be clustered, regardless of where they appear in the image. For and input data vector x;

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_z \end{bmatrix} \Rightarrow k - means\ on\ the\ elements \Rightarrow c_1, c_2 \dots c_k\ \ (k\ clusters)$$

This method is relatively cheap since the only complexity is related to finding the k centroids.

Spectral Clustering

At a high level, the theory behind spectral clustering involves clustering a graph into K, subgraphs. This is done by making cuts to separate the K highly connected groups. A measure of the clustering quality can be defined as the cost function,

$$\sum_{k=1}^{K} \frac{cut(V_k)}{size(V_k)}$$

Such that cut(V_k) is the total number of edges that connect to V_k to the rest of the graph, and size(V_k) is the number of vertices in V_k. Minimizing the cost function is a complex combinatorial problem, but can be achieved approximately through eigendecomposition and k-means.

The process for spectral clustering can be broken into 3 high level steps:

1) Creating the adjacency matrix, W, and the graph Laplacian, L
2) Computing the eigendecomposition of L to get the first K smallest eigenvectors.
3) Clustering the rows of the first k eigenvectors matrix into k clusters

Multiple methods exist for each step, so I will go in further detail about how I choose to do each step. My implementation follows the algorithm proposed by Ng, Jordan, and Weiss (2002), in the Luxburg paper (2007).

Step 1: Adjacency Matrix and graph Laplacian

Given and input data vector x, we can create a similarity function that relates x_i and x_j

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_z \end{bmatrix} \qquad S_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma_1^2} - \frac{distance(x_i, x_j)}{\sigma_2^2}\right)$$

The advantage of this similarity function is that it takes into account both the similarity between pixel intensities, as well as how close they are in the image. Additionally, the parameters σ_1 and σ_2 can be chosen to optimize the balance of weights.

For an image with z pixels, z^2 comparisons must be made and stored. This presents and issue, since storing and working with huge matrices is slow and costly. Thus, I implemented a k-nearest neighbor strategy that only compares pixels to other pixels within a certain radius. Thus, my adjacency matrix, W, can be defined as

$$W_{ij} = \begin{cases} S_{ij} & if\ distance(x_i, x_j) < R \\ 0 & Otherwise \end{cases}$$

Setting R to about half the width of an image results a W matrix that is about 50% full, so keeping R under this results in a somewhat sparse matrix

The clustering algorithm I chose uses the normalized graph Laplacian, which is defined as follows:

$$L_{sym} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

Where

$$D = diag(d) \quad and \quad d_i = \deg(i) = \sum_{j=1}^{n} W_{ij};\ i \neq j$$

This comes from the normalized cut objective,

$$Minimize\ Trace(X^T L X)$$
$$Subject\ to\ X^T D X = I$$

And let

$$X = D^{-\frac{1}{2}} Y$$

So the objective becomes

$$Minimize\ Trace\left(Y^T D^{-\frac{1}{2}} L D^{-\frac{1}{2}} Y\right)$$
$$Subject\ to\ Y^T Y = I$$

$$L_{sym} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

Computing the eigendecomposition of the normalized graph Laplacian

Once the normalized graph Laplacian is computed, the first K smallest eigenvectors $y_1, y_2 \dots y_k$ must be calculated. This is done using MATLAB's eigs() function.

The result is a matrix $U = [y_1, y_2 \dots y_k]$. The next step, according to the algorithm I used, is to normalize the rows of U to create a new matrix T. However, this solution is not necessarily an indicator matrix, which leads us to the final step, clustering the rows of T.

Clustering the rows of the first k eigenvector matrix into k clusters

This step is to round T to a valid indicator matrix. Using MATLAB's built in k-means function, we cluster the rows into K groups. This step introduces some randomness due to the initialization of the centroids, but for the most part my results were repeatable.

Once the rows of T are clustered, we can relate it back to the data vector. If T_i is the i-th row of T, and $g(T_i)$ is the group associated with the i-th row of T, then

$$g(x_i) = g(T_i)$$

From this, we can construct a segmented image.

Results:

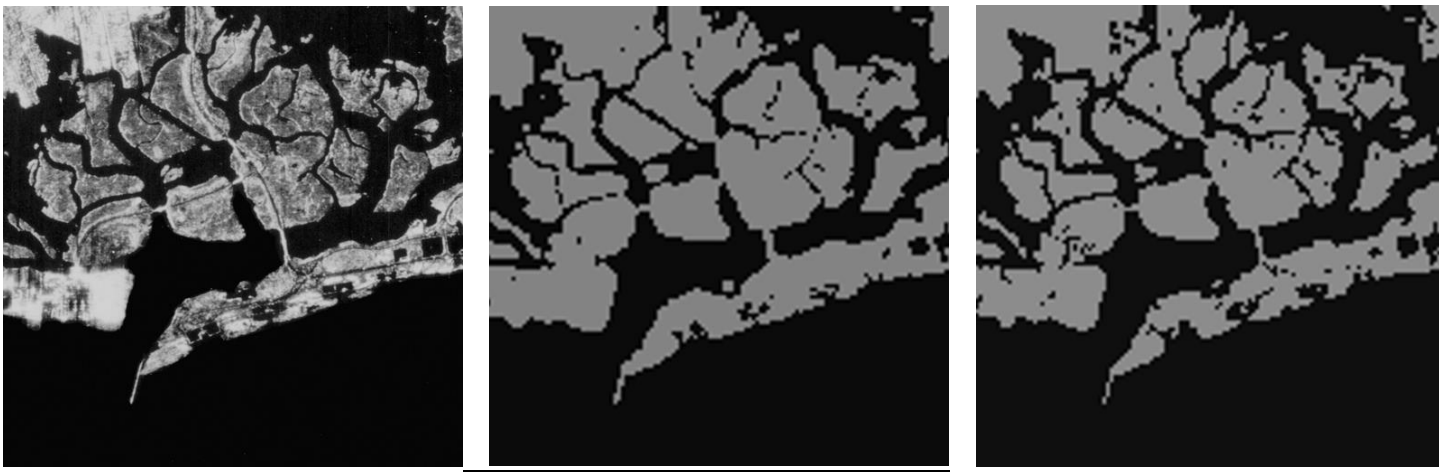
Each image was scaled to a 128x128 resolution to keep processing times manageable. At this scale, the spectral clustering took in the order of tens of minutes to compute. Each SAR image highlights some good or bad aspect of both methods. I will elaborate on this on the analysis section.

Figure (1) Long Island West (a)

a) **b)** **c)**

a) Original image b) Results of spectral clustering, k = 2 c) Results of k-means, k = 2

Figure(2) Long Island West (b)



a) **b)** **c)**

a) Original image b) Results of spectral clustering, k = 2 c) Results of k-means, k = 2

Figure (3) Washington

a) **b)** **c)**

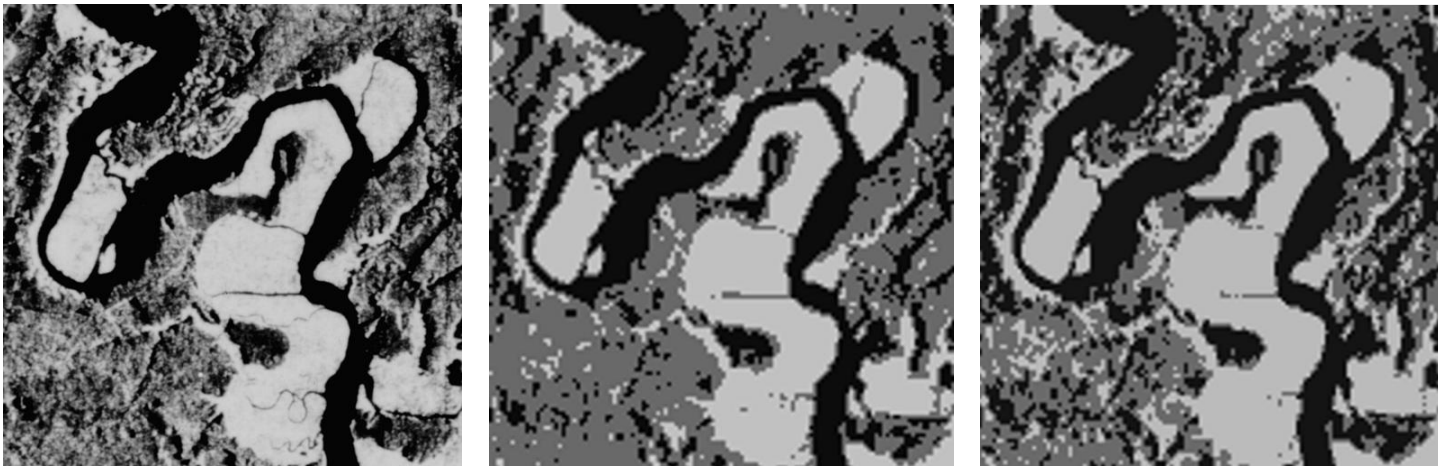
a) Original image b) Results of spectral clustering, k = 2 c) Results of k-means, k = 2

Figure (4) Port Moller

a) **b)** **c)**

a) Original image b) Results of spectral clustering, k = 3 c) Results of k-means, k = 3

Figure (5) Williamsburg



a) **b)** **c)**

a) Original image b) Results of spectral clustering, k = 3 c) Results of k-means, k = 3

Analysis:

Since these images are real images obtained from USGS, I don't have a ground truth to evaluate the results. However, I can qualitatively discuss the results based on a well defined criteria. This criteria is defined as follows

1) Accuracy – how well the clusters pick up finers details on the image
2) Correctness – how well the clusters segment the image into clearly different parts

Each figure will be evaluated to determined the benefits and drawbacks of both methods.

Figure (1):

This image was chosen because it clearly represents two clusters, land, and water. The image is not very complicated, so both methods were fairly accurate. However, K-means picked up slightly finer details around the shoreline. Spectral clustering was more correct because there are less “ocean” pixels on land. K-means mis-identified quite a few pixels on land because they are dark in the original image. However, these pixels are far away from the majority of dark pixels. This means they will be less connected in the spectral clustering method, classifying most as land.

Figure (2):

This image also contains only land and water, but the geography is much more complicated. Unsurprisingly, spectral clustering misses out on some of the finer details that k-means picks up. The number of mis-identified pixels are slightly more in the k-means, but the difference is not huge. In this case, the land mass does not contain many dark pixels, so k-means does not run into the same problem as before. Both methods are about equally correct, identifying land as land and water as water.

Figure (3):

Again, this image contains only land and water. However, the land surface is highly irregular, and contains many dark pixels. Right away we can see that k-means has identified a large number of land pixels as water. Spectral clustering still had some problems, but did much better overall in correctness. When it comes to accuracy, both methods accurately depicted details about the riverbank/shoreline.

Figure (4):

This image was chosen for its high contrast, and clear regions. It appears that the image should be clustered into snow covered land, beach/shore, and water. We can see in the original image that each region is relatively homogenous. Unsurprisingly, k-means and spectral clustering performed well. Differences in accuracy and correctness are negligible. However, its worth mentioning that k-means takes much less time.

Figure (5):

This image also has 3 clusters, representing water, mud/riverbank, and land with vegetation. The vegetation makes the land surface irregular, and presents challenges for both methods. K-means accurately depicted the geographical features, but suffered greatly in the number of mis-identified pixels. Much of the vegetation was classified as water, even if it was far inland. However, spectral clustering had equal accuracy with far better correctness. Some vegetation that was close to the riverbank was mis-identified, but the vegetation far away was correctly identified.

Conclusion:

Both methods produce good results in certain situations. As shown in the previous examples, k-means tends to work well when the image has few irregularities/noise. Since k-means has no spatial parameters, it will identify pixels only based on intensity. K-means also picks up finer details in almost all situations. Perhaps the greatest advantage of k-means is the low complexity, and thus fast computation time.

Spectral clustering is admittedly very slow in comparison. Computing a 128x128 image could take as long as 25 minutes, with most of the time dedicated to eigendecomposition. However, the correctness results are always equal or better than k-means. Spectral clustering especially shines when the image has irregularities and noise. It is also worth mentioning human effort in each method.

K-means is truly unsupervised, and does not require tweaking parameters to get results. On the other hand, Spectral clustering requires tweaking of the σ_1 and σ_2 parameters. The results are highly dependent on these

parameters, and I could find a good method other than trial and error. The goal is to connect pixels that are close in intensity, as well as proximity. It seems like favoring distance more works well in noisy images (fig. 3), and favoring intensity works better on high contrast homogenous images (fig. 4). Nevertheless, these parameters make finding a good segmentation on the first try difficult.

It seems like k-means is a simple method that works well on certain high contrast images. However, spectral clustering is more robust, and with some time and effort, can produce much better results. That being said, the time and effort may not be worth it. The choice of method is very situation, which is why we have smart humans to make the decision.

MATLAB Code:

All my code can be found at this link:

https://github.com/giacomofratus/spec_clust

References:

Luxburg, U., A Tutorial on Spectral Clustering, (2007*), Statistics and Computing*, 17 (4).