14 Spectral segmentation and normalized cuts

Spectral clustering is an approach to data clustering problem, and it includes a number of related techniques. Spectral clustering is used in machine learning, computer vision and signal processing, with applications in processing speech spectrograms, DNA gene expression analysis, document retrieval and computation of Google page rank. The name *spectral* originates from the mathematical term *spectrum* (a set of the egenvalues of a given matrix), and this is because spectral clustering utilizes eigenvalues and eigenvectors of the data similarity matrix.

Spectral clustering is one of the fundamental data clustering approaches, it is easy to implement and solve by standard linear algebra software, there is no assumptions on the nature of the clusters, and the techniques have been mathematically rigorously proved. Disadvantages include high computational and memory requirements of the direct implementation. For this reason the practical use of spectral clustering often involves computational simplifications and significant pre- or postprecessing.

Spectral approach has gained a great popularity for image segmentation following the seminal paper on normalized cuts by Shi and Malik [1]. Recent uses of spectral approach is in the superpixel segmentation. Another intriguing example is the Copenhagen-based company Spektral (formerly known as CloudCutout) where spectral segmentation is a part of the successful green-screen removal product Figure 14.1.

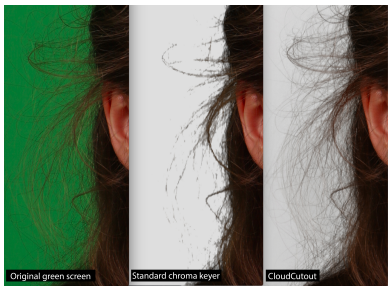

Figure 14.1: Spektral (formerly known as CloudCutout) green-screen product.

Spectral methods can be applied to the data which is represented using a similarity matrix, and is often described in terms of graph partitioning. For a comprehensible coverage of (general) spectral clustering we recommend an excellent tutorial by von Luxburg [2]. We will here briefly cover spectral clustering, and will then turn to its use in image segmentation.

Boiled down to four words, the essence of spectral clustering is: *Eigensolution gives graph partitioning*. To be able to understand spectral

[1] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000

[2] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17 (4):395–416, 2007

clustering, you need to be acquainted with the concept of graph cuts in order to describe the problem we want to solve. Furthermore, we need to represent a graph using an adjacency matrix and a closely related Laplacian matrix. Then we can see how eigensolution provides a solution to a graph cut problem.

14.1 *Graph cuts, graph representations and eigensolutions*

Recall that a graph consists of nodes and edges, and in general may be node-weighted, edge-weighed, directed or undirected. In context of spectral image segmentation, each pixel will correspond to a graph node, and pairs of pixels define graph edges – we will get back to image segmentation after covering the general case. For spectral clustering we work with edge-weighted undirected graphs which we represent using an adjacency matrix W with elements w_{ij} being the weight of the edge connecting the node i and a node j. Consider for example a graph in Figure 14.2 consisting of 12 nodes. This graph can be represented in terms of a 12×12 adjacency matrix, as illustrated in Figure 14.3.

A graph cut is a partitioning of a graph. The graph partitioning problems are concerned with finding a graph cut with the least cost. The simplest way of defining the cost of a cut is to consider all edges between the two partitions

$$\text{cut}(A, B) = \sum_{i \in A, j \in B} w_{ij} \,,$$

but that might lead to unbalanced cuts. For this reason we might prefer using some other measure of the cut cost, which also consider the size of the partitions. Commonly used are normalized cuts

$$\text{Rcut}(A, B) = \frac{\text{cut}(A, B)}{|A|} + \frac{\text{cut}(A, B)}{|B|} \,,$$

where we use the notation $|A|$ for a number of vertices in subset A, but one could also consider ratio cut and min-max cuts

$$\text{Ncut}(A, B) = \frac{\text{cut}(A, B)}{\text{vol}(A)} + \frac{\text{cut}(A, B)}{\text{vol}(B)} \,,$$

$$\text{MMcut}(A, B) = \frac{\text{cut}(A, B)}{\text{cut}(A, A)} + \frac{\text{cut}(A, B)}{\text{cut}(B, B)} \,,$$

where $\text{vol}(A)$ is weight of all edges associated with the subset, i.e. $\text{vol}(A) = \sum_{i \in A} d_i$ and $d_i = \sum_j w_{ij}$ is a degree of ith node. Figure 14.4 shows three graph cuts, while Figure 14.5 lists the costs associated with the three cuts given by different measures. You may confirm that the values are correct, and notice the different balancing properties of the cost measures.

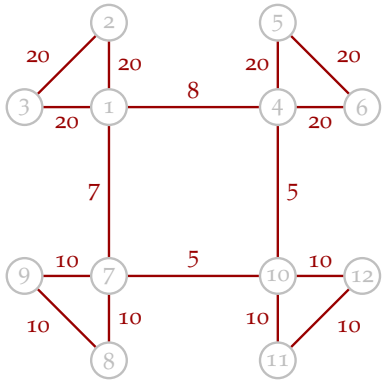

Figure 14.2: An example of an edge-weighted undirected graph.

	1	2	3	4	5	6	7	...
1	0	20	20	8	0	0	7	...
2	20	0	20	0	0	0	0	...
3	20	20	0	0	0	0	0	...
4	8	0	0	0	20	20	0	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Figure 14.3: An adjacency matrix of a graph shown in Figure 14.2.

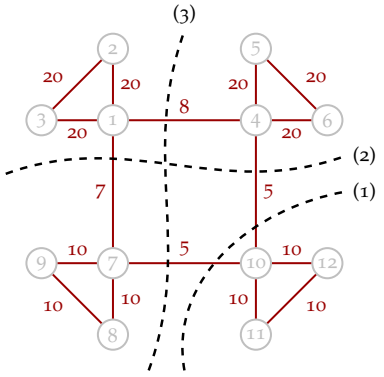

Figure 14.4: Three different partitioning of a graph.

	(1)	(2)	(3)
cut	10	12	13
Rcut	4.4̇	4.0	4.3̇
Ncut	0.1723	0.1293	0.1268
MMcut	0.3939	0.2784	0.2709

Figure 14.5: Values of the different cuts shown in Figure 14.4.

It turns out that the solution to the graph cut problem may be found by considering the eigensolution of the matrix closely related to the adjacency matrix – the graph Laplacian. Depending on the cost measure, the derivation will be slightly different, leading to the different (unnormalized and normalized) versions of the graph Laplacians. To normalize the Laplacian, we first define a diagonal degree matrix $\mathbf{D}$ with diagonal elements being node degrees $d_i = \sum_j w_{ij}$.

We define unnormalized graph Laplacian

$$\mathbf{L} = \mathbf{D} - \mathbf{W},$$

and two normalized graph Laplacians

$$\mathbf{L}_{\text{sym}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2},$$

$$\mathbf{L}_{\text{rw}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}.$$

These matrices are closely related to each other, and so are their spectra. All three matrices have real-valued eigenvalues with 0 being the smallest eigenvalue, see tutorial by von Luxburg for a more complete list of properties.

For our purposes, the most important are eigenvectors of $\mathbf{L}_{\text{rw}}$, i.e. vectors satisfying $\mathbf{L}_{\text{rw}}\mathbf{u} = \lambda\mathbf{u}$. The smallest eigenvectors (corresponding to smallest eigenvalues) yield solution relaxed versions of finding the normalized cut. The relaxed solution is subsequently transformed into an approximate discrete solution to the original problem, for example using k-means clustering. In particular, the second smallest eigenvector gives the partitioning of the data in two subsets – the very smallest eigenvector (corresponding to 0) is constant.

It can be shown that eigenvectors of $\mathbf{L}_{\text{rw}}$ are generalized eigenvectors of $\mathbf{L}$ and $\mathbf{D}$, see againg von Luxburgs tutorial, Proposition (3) part 3. Therefore, to fint the solution to normalized cut, one may also seek solution to generalized eigenproblem $\mathbf{L}\mathbf{u} = \lambda\mathbf{D}\mathbf{u}$. This is the formulation of normalized spectral clustering according to the original paper by Shi and Malik. In practice, solving a standard eigenproblem requires less computation.

We can utilize other matrix algebra identities to make computation of the spectra more accurate and/or efficient. Many eigensolvers are more accurate when working of symmetric matrices. A strategy exploiting matrix symmetry would involve computing eigenvectors of the (symmetric) $\mathbf{L}_{\text{sym}}$, and transforming those to eigenvectors of $\mathbf{L}_{\text{rw}}$ by multiplying with $\mathbf{D}^{-1/2}$, see tutorial by von Luxburg, Proposition (3) part 2. Furthermore, if only a subset of eigenvectors is to be found, some eigensolvers are more efficient when finding eigenvectors corresponding to largest eigenvalues. This may be utilized by noticing that that smallest eigenvectors of $\mathbf{I} - \mathbf{A}$ are largest eigenvectors of $\mathbf{A}$.

14.2 *Clustering 2D points*

You should implement two functions. One function should take and affinity matrix and return eigenvectors corresponding to solution for normalized cuts. The second function should perform discretization of the eigenvectors using k-means.

You should first test your implementation on a small 2D point set. You can use points previously used for neural networks, but we also provide five point sets in a mat file `points_data.mat`. For each of the point sets you should:

- Visualize the point set, and identify the clusters (there are 2 clusters in set 3 and 5, and 3 clusters in set 1, 2 and 4).
- Construct the affinity matrix W. Use the fully connected graph and Gaussian similarity function (Luxburg, Section 2.2). You should initially estimate parameter σ so that it reflects the distance between the neighbouring points of the point cloud.
- Compute eigenvectors and clustering given by the normalized cut. Visualize the clustering.
- Determine ordering (permutation) of the points according to the clustering (so that points from the first cluster come first, followed by points in the second cluster, etc.)
- Visualize the values of the second eigenvector, first for unsorted points, then for sorted points.
- Visualize affinity matrix, and the affinity matrix for sorted points.
- Estimate the parameter σ which results in a meaningful clustering. You will need to change the parameter σ between point sets.

14.3 *Image segmentation*

Now we use spectral clustering on a pixels of a small image. Consider some of the provided test images. You might want to (drasticaly!) reduce the size of your images, to avoid memory problems.

You should:

- Construct the affinity matrix W using Equation (11) from article by Shi and Malik. Initially estimate parameters σ_I, σ_X and r. Instead of setting a radius r, for our small example you may use a fully connected graph (i.e. ignore if–otherwise condition of Equation (11) which sets affinity of distant points to 0).
- Visualize the spatial part of W, the brightness part of W and the final W.

- Compute eigenvectors and clustering given by the normalized cut.
- Visualize the values of the second eigenvector on the image grid.
- Visualize the segmentation results.
- Estimate the model parameters to obtain meaningful segmentation.

Start by the grayscale image. Use 2 clusters for plane and 5 clusters for vegetables. For the similarity (brightness, color) part of W treat an RGB value of each pixel as a vector to compute the Euclidian distance between the pair of pixels. Try also clustering the pixels using k-means clustering by treating RGB pixels values as vectors.

Consider adapting spectral methods to be able to handle larger images.