Approaches to Brain Parcellation using Energy Statistics and Graph Partitioning

Felix Xiao

March 4, 2016

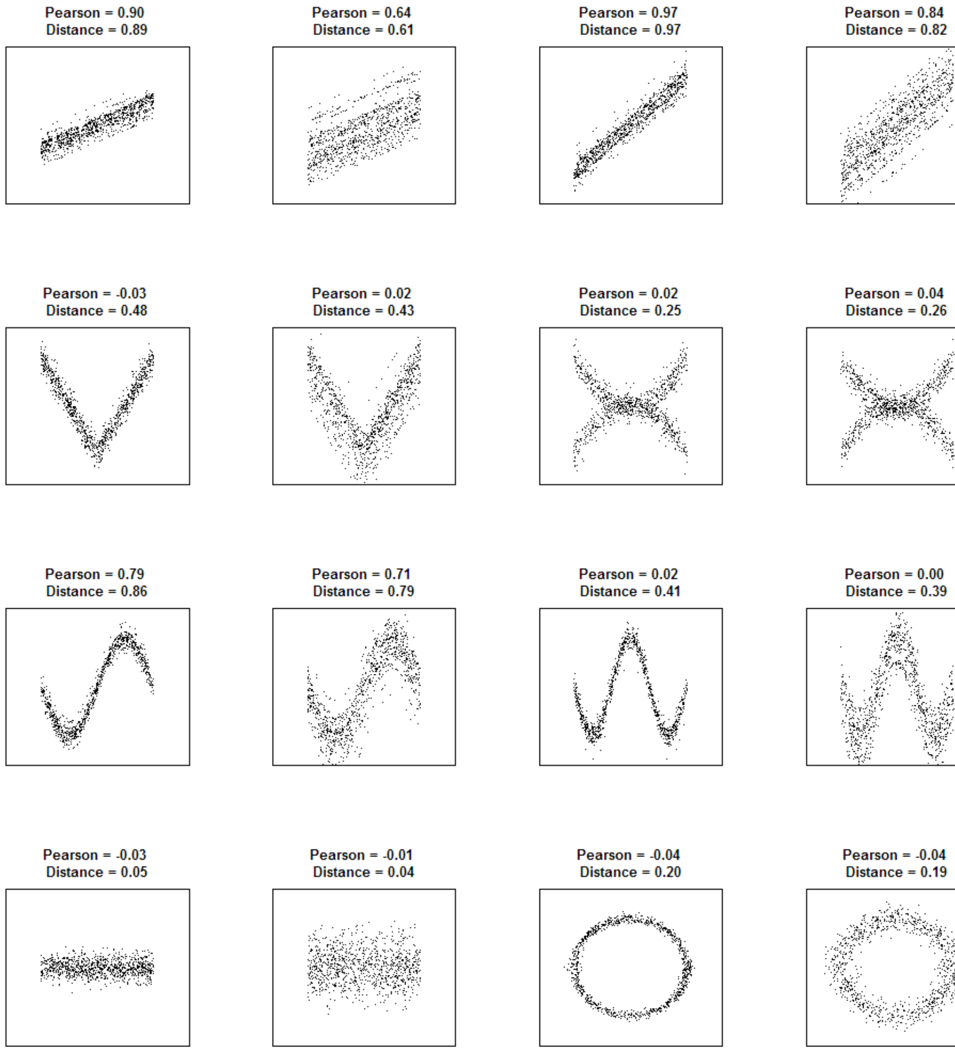
Contents

Chapter 1

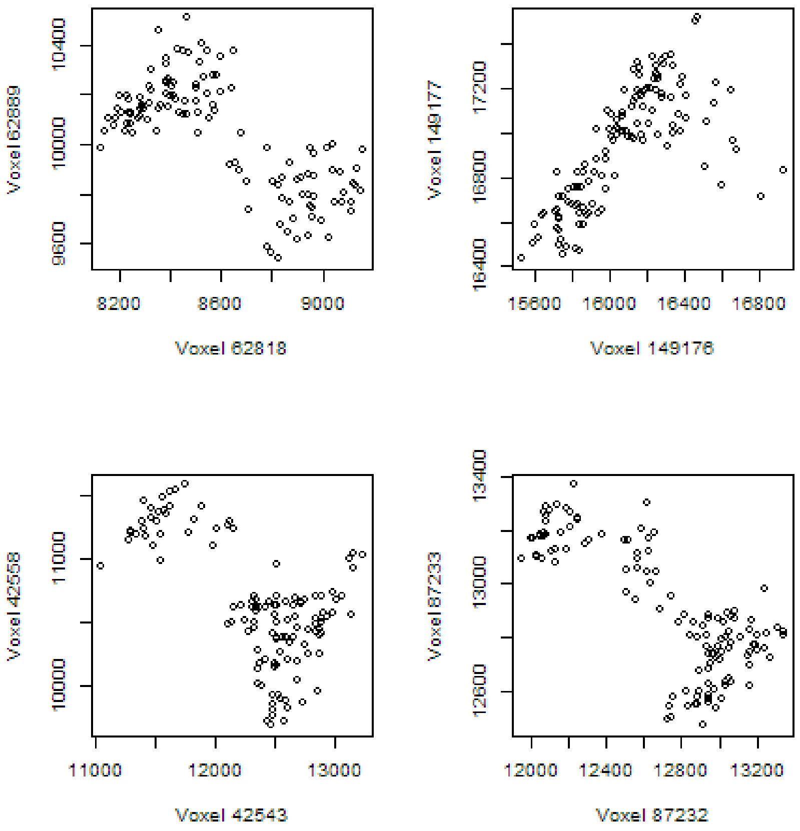
Functional MRI Data and Brain Parcellation

Functional parcellation of the human brain can be defined as the problem of partitioning the voxels into k disjoint connected components with the goal that the voxels within each component are, in a rough sense, s̈imilarẗo each other and voxels in different components are less s̈imilar̈. Such similarity has been defined in a multitude of ways in the literature [see lit review section ...]. For this project thus far I have taken similarity between voxels to mean statistical dependence.

To measure dependence, statisticians have traditionally used the Pearson correlation coefficient, in addition to the rank-based Kendall tau and Spearman rho. These statistics work well when the underlying relationship between the two random variables is linear, in the case of Pearson, or can be linear after a monotonic transformation, in the case of Kendall and Spearman. Due to their restrictions, these correlation coefficients will fail to capture many kinds of dependency relationships. The figure below illustrates several instances of pairs of random variables whose depencency structure is not detected by the three correlation coefficients.



Non-linear dependency relationships also exist in the ABIDE 50002 fMRI data. The scatterplots below show time samples of spatially adjacent voxels. These instances were found by searching for the maximum difference in rank of energy distance correlation and the coefficient of determination, or Pearson squared.



Many studies on functional parcellation (Craddock 2012; Bellec 2006; Heller 2006) use Pearson's coefficient as the similarity measure between nearby voxels. Apart from underestimating the important of non-linear relationships, this method also distinguishes positive, upward-sloping correlation from negative. As a result in many of the edges between different parcels, the corresponding voxels would be strongly dependent with negative correlation.

In this investigation, all parcellation and validation procedures were conducted on the ABIDE 50002 fMRI data set. This data set contains 233305 voxels and 124 time samples. Spatial information is encoded as a graph; each voxel is represented by a vertex, and each vertex has up to 6 edges connecting the voxel to its cubically adjacent neighbors. The weights on the edges are sample energy distance correlations between the two connected voxels (Szekely 2013).

Chapter 2

Energy Statistics

2.1 Energy Covariance

For some positive weight function $w : \mathbb{R}^p \times \mathbb{R}^q \mapsto [0, \infty)$ define the norm $\| \cdot \|_w : \{\gamma : \mathbb{R}^p \times \mathbb{R}^q \mapsto \mathbb{C}\} \mapsto [0, \infty)$ as

$$\|\gamma\|_w^2 = \int_{\mathbb{R}^{p+q}} |\gamma(s,t)|^2 w(s,t) dsdt$$

Definition 2.1.1 *(Distance covariance). Let X and Y be two d-dimensional random vectors with $\mathbf{E}\|X\| + \mathbf{E}\|Y\| < \infty$. Their distance covariance is*

$$\begin{aligned} \mathcal{V}^2(X,Y) &= \|\varphi_{X,Y}(s,t) - \varphi_X(s)\varphi_Y(t)\|_w^2 \\ &= \int_{\mathbb{R}^{p+q}} \frac{|\varphi_{X,Y}(s,t) - \varphi_X(s)\varphi_Y(t)|^2}{\|s\|^{1+p}\|t\|^{1+q}} dsdt \end{aligned}$$

where $w(s,t) = \dfrac{1}{\|s\|^{1+p}\|t\|^{1+q}}$.

It is clear that $\mathcal{V}^2(X,Y) = 0 \iff X \perp\!\!\!\perp Y$.

Proposition 2.1.2

$$\begin{aligned} \mathcal{V}^2(X,Y) &= \mathbf{E}[\|X - X'\|\|Y - Y'\|] + \mathbf{E}[\|X - X'\|]\mathbf{E}[\|Y - Y'\|] - 2\mathbf{E}[\|X - X'\|\|Y - Y''\|] \\ &= \mathrm{Cov}(\|X - X'\|, \|Y - Y'\|) - 2\mathrm{Cov}(\|X - X'\|, \|Y - Y''\|) \end{aligned}$$

Proof.

Definition 2.1.3 *(Distance variance).*

$$\mathcal{V}^2(X) = \mathcal{V}^2(X,X)$$

Definition 2.1.4 *(Distance correlation).*

$$\mathcal{R}^2(X,Y) = \frac{\mathcal{V}^2(X,Y)}{\mathcal{V}(X)\mathcal{V}(Y)}$$

For iid sample realizations $\{(X_i, Y_i)\}_1^n$, let $\widehat{\varphi_X}(t) = \frac{1}{n}\sum_{i=1}^n e^{it^T X_i}$ be the empirical characteristic function for X and likewise for Y. An estimate of $\mathcal{V}^2(X,Y)$ replaces the unknown characteristic functions with the empirical characteristic functions.

Proposition 2.1.5

$$\widehat{\mathcal{V}}^2(X,Y) \equiv \|\widehat{\varphi_{X,Y}}(s,t) - \widehat{\varphi_X}(s)\widehat{\varphi_Y}(t)\|_w^2 = S_1 + S_2 - 2S_3$$

where $w(s,t)$ as above and

$$\begin{aligned} S_1 &= \frac{1}{n^2}\sum_{k=1}^n\sum_{l=1}^n \|X_k - X_l\|\|Y_k - Y_l\| \\ S_2 &= \left(\frac{1}{n^2}\sum_{k=1}^n\sum_{l=1}^n \|X_k - X_l\|\right)\frac{1}{n^2}\sum_{k=1}^n\sum_{l=1}^n \|Y_k - Y_l\| \\ S_3 &= \frac{1}{n^3}\sum_{k=1}^n\sum_{l=1}^n\sum_{m=1}^n \|X_k - X_l\|\|Y_k - Y_m\| \end{aligned}$$

Alternatively, we can let $A, B \in \mathbb{R}^{n\times n}$ such that $A_{kl} = \|X_k - X_l\|$ and $B_{kl} = \|Y_k - Y_l\|$ (A and B are symmetric elementwise nonnegative). Let $\overline{X} = \frac{1}{n^2}\sum_{k,l=1}^n X_{kl}$. Then

$$\hat{\mathcal{V}}^2(X,Y) = \overline{A \circ B} + \overline{A}\cdot\overline{B} - \frac{2}{n}\overline{(AB)}$$

where $\circ$ means element-wise multiplication.

Estimates of distance variance and distance correlation are defined analogously.

Proposition 2.1.6

$$\mathcal{V}(v_1 + a_1 Q_1 X, v_2 + a_2 Q_2 Y) = \sqrt{|a_1 a_2|}\mathcal{V}(X,Y)$$

Definition 2.1.7 *(α-distance covariance). For $0 < \alpha < 2$*

$$\mathcal{V}_\alpha^2(X,Y) = \frac{1}{C(p,\alpha)C(q,\alpha)}\int_{\mathbb{R}^{p+q}} \frac{|\varphi_{X,Y}(s,t) - \varphi_X(s)\varphi_Y(t)|^2}{\|s\|^{\alpha+p}\|t\|^{\alpha+q}} dsdt$$

Proposition 2.1.8 *If $\mathbf{E}[\|X\|^\alpha] + \mathbf{E}[\|Y\|^\alpha] < \infty$ then*

$$\mathcal{V}_\alpha^2(X,Y) = \mathbf{E}[\|X-X'\|^\alpha\|Y-Y'\|^\alpha] + \mathbf{E}\|X-X'\|^\alpha\mathbf{E}\|Y-Y'\|^\alpha - 2\mathbf{E}[\|X-X'\|^\alpha\|Y-Y''\|^\alpha]$$

Corollary 2.1.9 *For $\alpha = 2$, $p = q = 1$, the distance correlation is the absolute value of Pearson's correlation coefficient.*

Chapter 3

Criteria for Evaluating Parcellations

In chapter one we discussed our graphical approach to the brain parcellation problem. We construct a weighted undirected graph where each vertex is a voxel. The graph reflects the spatial position of the voxels; it connects each vertex to the vertices representing the voxel's six cubically adjacent neighbors. The weights on these edges are sample energy distance correlation statistics between the adjacent voxels in the time series and they measure statistical dependence between the voxels. Let $G(V, E)$ denote this graph, its vertices, and its edges.

In this context, a valid k-fold partition $\mathcal{P}_k$ of the graph G is a collection of vertex subsets $(V_1, ..., V_k)$ satisfying the following:

1. $V_i \neq \varnothing$ for all $V_i \in \mathcal{P}_k$
2. $\bigcup\limits_{i=1}^{k} V_i = V$
3. $V_i \cap V_j = \varnothing$ for all $V_i, V_j \in \mathcal{P}_k$
4. V_i is connected (i.e. for every two vertices in V_i, there is a path between them) for all $V_i \in \mathcal{P}_k$

In this chapter we will suggest various criteria for measuring the goodness-of-fit of partitions. We will argue why these criteria are sensible from a neuroscience perspective.

3.1 Within-Parcel Similarity

Voxels in the same parcel are ideally highly dependent on one another in the time series of fMRI data. As discussed in the previous chapter, distance correlation is a good measure of dependence. The distance correlation between two random vectors equals zero if and only if the two random vectors are independent, which is not true of correlation statistics such as Pearson's.

Let $\mathcal{R}(X, Y)$ denote the distance correlation between two voxels X and Y. Let $\mathcal{P} = \{P_1, ..., P_k\}$ be a k-fold partition. We define the following criterion

Definition 3.1.1 *Within-Score.*

$$\frac{1}{k} \sum_{P \in \mathcal{P}} \frac{1}{|P|^2} \sum_{X,Y \in P} \mathcal{R}(X, Y)$$

The Within-Score is non-spatial; it considers all pairs of voxels equally regardless of whether they are adjacent. As a result, it is a good measure of how much the voxels within each parcel are dependent on each other as a set. The downside of this criterion is that it is very expensive to compute. With over 300,000 voxels in an fMRI data set we would potentially have to compute tens of billions of distance correlation statistics if the number of parcels is small.

There are two solutions to this. One is to subsample: for every parcel, compute the distance correlation matrix for a small subset of the voxels in the parcel and construct a confidence interval around an estimate of the Within-Score. Another solution is to reduce the image resolution: merge multiple adjacent voxels into a single voxel by averaging the time series and then compute the Within-Score.

An alternative and far less expensive criterion that measures within- parcel similarity works by counting distance correlations between adjacent pairs of voxels. For some parcel P let $E_P = \{(i, j) \in E : i \in P \text{ and } j \in P\}$.

Definition 3.1.2 *Adjacent-Score.*

$$\frac{1}{k} \sum_{P \in \mathcal{P}} \frac{1}{|E_P|} \sum_{(X,Y) \in E_P} \mathcal{R}(X, Y)$$

Rather than treat parcels as sets with no spatial information, the Adjacent-Score does the opposite by only considering the pairwise dependency of adjacent voxels.

Other possibilities that we did not explore are considering all pairs of voxels up to some maximum spatial distance from each other and performing a weighted averaging of sample pairwise distance correlations, with weights that depend on spatial distance.

3.2 Between-Parcel Dissimilarity

To evaluate parcellation quality, it is also useful to measure how dependent voxels belonging to different parcels are on each other. To this end we define two criterion similar to the Within-Parcel criterion; a non-spatial metric called the Between-Score and a spatial metric called the Boundary-Score. For two different parcels P and P', let $E_{P,P'} = \{(i, j) \in E : i \in P \text{ and } j \in P'\}$.

Definition 3.2.1 *Between-Score.*

$$\frac{2}{|\mathcal{P}|(|\mathcal{P}|-1)} \sum_{P,P' \in \mathcal{P} P \neq P'} \frac{1}{|P||P'|} \sum_{X \in P Y \in P'} \mathcal{R}(X,Y)$$

Definition 3.2.2 *Boundary-Score.*

$$\frac{2}{|\mathcal{P}|(|\mathcal{P}|-1)} \sum_{P,P' \in \mathcal{P} P \neq P'} \frac{1}{|E_{P,P'}|} \sum_{(X,Y) \in E_{P,P'}} \mathcal{R}(X,Y)$$

Chapter 4

Local Search and Graph Growing Heuristics

We introduce several algorithms for generating brain parcellations. The algorithms in this chapter are all local search heuristics; they begin with N unconnected vertices and iteratively join adjacent ones into components until some stopping criterion is met.

For each algorithm, the resulting parcellation is presented, discussed, and evaluated according to the criteria introduced in the previous chapter.

4.1 Unconstrained Add-Edge

The first and simplest algorithm starts with an empty graph of N vertices and sequentially adds edges between adjacent voxels in order of highest sample distance correlation, until the graph has some pre- specified number of connected components K.

We will refer to this algorithm as Ünconstrained Add-Edge:. A naive implementation of would re-compute the number of connected components in the graph (using linear-time bread-first or depth-first search) after each addition of an edge, resulting in a costly $O(EN)$ time complexity. A more efficient implementation takes advantage of the fact that each addition of an edge decreases the number of components in the graph by at most 1. Hence the algorithm needs only to compute the number of connected components after adding $k - K$ edges, where k is the current number of connected components of the graph, beginning at N.

```
k := N
i := 1
while k > K
    repeat k - K times
        add the ith highest-weighted edge to the graph
        i := i + 1
    end
    k := compute number of connected components
end
```

Another implementation uses a binary search-type strategy and is $O((N+E)\log E)$. The idea is to ̈searchf̈or the last edge to add to the graph by maintaining a range of possible last edges. In each iteration, the algorithm would add to the graph edges 1 to the midpoint of this range, compute the number of connected components, and adjust the range based on whether the number of components is higher or lower than the target K.

```
l := 1
h := E
repeat
    m := (l + h) / 2
    add edges from 1 to m to an empty graph
    k := compute number of connected components
    if k = K
        done
    else if k < K
        h := m
    else if k > K
        l := m
    end
end
```

The Unconstrained Add-Edge algorithm produces severely imbalanced parcellations. In the 100-component graph, there was one component containing over 99.9% of all the vertices in the graph. The following algorithm introduces a modification that address this issue.

4.2 Size-Constrained Add-Edge

The Size-Constrained Add-Edge algorithm works in a similar manner to the Unconstrained version, adding edges to the graph in decreasing order of distance correlation. The Size-Constrained version differs by applying a filter to each edge considered, adding the edge only if at least one of the two following conditions are met:

1. At least one of the two components bridge by the edge is of size less than some prespecified parameter $s_{\min}$.
2. The union of the two components is of size $\leq s_{\max}$.

Letting K denote the target number of components in the graph, the Size-Constrained Add-Edge algorithm can be written as:

```
sort edges in decreasing energy correlation order
k := N, number of components
for e = 1, ..., E
    (i, j) := vertices of edge e
    I := component containing i
    J := component containing j
    if I = J
        continue
    else if (size(I) < s_min or size(J) < s_min)
            or size(I + J) <= s_max
        add e to the graph
        k := k - 1
        if number of components = K
            break
        end
    end
end
```

The naive implementation must use BFS/DFS in each iteration to compute the size of components I and J, and hence must have time complexity $O(EN)$. Fortunately, there is a way to sub-linearly update information on the components of the graph, using the union-find data structure.

4.2.1 Union-Find

The core Union-Find data structure begins with an empty graph of N vertices and supports two operations. union(i, j) adds an edge between vertices i and j. root(i) returns an identifier for the component to which vertex i belongs. All vertices in the same component have the same root. We modified Union-Find to support an additional operation. component_size(i) returns the number of vertices belonging to the component containing i.

Union-Find represents each component as a rooted tree, with vertices in the graph mapping to nodes in the tree. Information about the tree is stored in two arrays of length N, parent and size, which are subject to the following invariants.

1. For each node i, parent[i] = node i's parent on the tree, unless i is a root node. If i is a root node, then parent[i] = i.

2. Nodes i and j are in the same component if and only if they are in the same tree, if and only if they share the same root node.

3. If i is a root node, then size[i] = the size of the component, or the number of nodes in the tree. If i is not a root node, then size[i] can be anything.

A baseline implementation of the three functions is

```
function root(i)
    while parent[i] != i
        i := parent[i]
    end
    return i
end

function union(i, j)
    parent[root(j)] := root(i)
end

function component_size(i)
    return size[root(i)]
end
```

In addition to the baseline code above, there are two important optimizations:

1. Weighted union maintains information of the sizes of each component so that the root of the smaller component always becomes a child of the larger component's root.
2. Path compression flattens the tree with each call to root. Specifically, when root is called on node i, each node traversed from i to the root has its parent set to be the root.

With these two optimizations, the time complexity of root, union, and component_size was proven in (Hopcroft 1973) to be at least as good as $O(\log^* N)$ where $\log^*$ is the iterated logarithm, defined as the number of times the natural log must be applied to N so that it becomes less than or equal to 1.

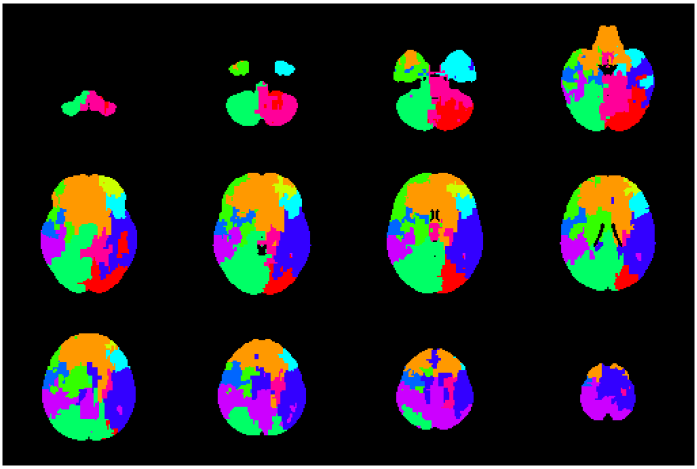
4.2.2 Results of Size-Constrained Add-Edge Parcellation

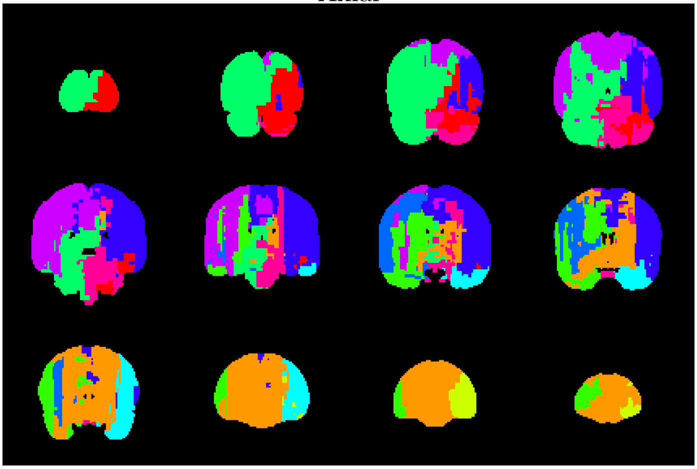
We ran Size-Constrained Add-Edge on the distance correlation graph and on a copy of the graph with randomized edge weights. We used parcellation criteria discussed in chapter 3 to evaluate the quality of the two resulting parcellations on the fMRI data used to generate the distance correlation graph. The results of the in-sample evaluation are shown in the table below:

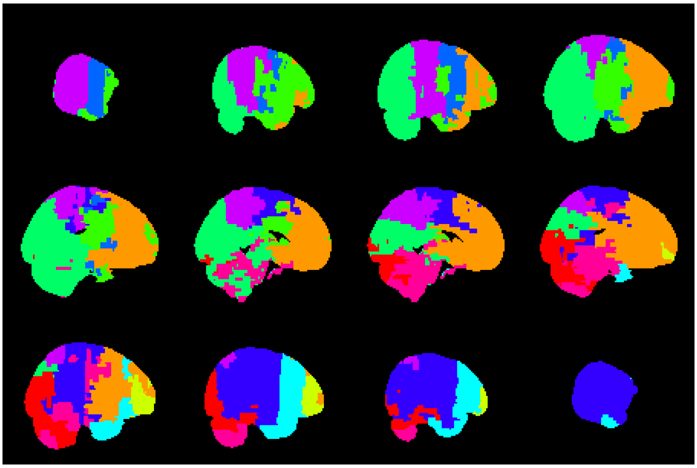
$s_{\min}$	$s_{\max}$	Within	Adjacent	Between	Boundary
1000	15000	0.314	0.726	0.296	0.521
1000	10000	0.313	0.719	0.306	0.517
1000	7500	0.328	0.721	0.313	0.514
1000	5000	0.320	0.727	0.303	0.516
1500	10000	0.318	0.711	0.315	0.518
750	10000	0.313	0.721	0.302	0.521
500	10000	0.318	0.718	0.307	0.521
	Random	0.304	0.705	0.295	0.719

The criterion that has seen the most significant improvement compared with the random graph parcellation is the Boundary-Score, with a decrease from 0.719 in the random graph parcellation to the range of 0.514 - 0.521. The Within- and Adjacent-Scores saw smaller but still noticeable improvements, and Between-Score saw no improvement from the random baseline.

We display the parcellation obtained from setting parameters $s_{\min} = 1000$ and $s_{\max} = 7500$ below. While the sizes of the parcels are much more balanced than the result of the Unconstrained Add-Edge, there are still noticeable differences in parcel size.


Axial


Coronal



Sagittal

4.3 Edge Contraction

The Edge-Contraction algorithm attempts to address two problems of the Size-Constrained Add-Edge algorithm: poor Adjacent-Score relative to randomized graph and unbalanced parcels. We hypothesized that one reason for a relatively low Adjacent-Score might be the following scenario: when a vertex is added to a component, it might have multiple edges to that component. One edge might have a very high weight; this is the one that is officially äddedẗ However, the other edges with far lower weights are implicitly added as well, lowering the average edge weights within the component.

The Edge-Contraction algorithm handles this issue by maintaining that there can be at most one edge between any two components A and B, and further that the weight on such an edge is the mean of the weights on all edges that connect a vertex in A with a vertex in B.

The graph starts out with N components, each a single vertex. In every iteration

1. The smallest component with the largest edge to another component is selected.

2. This largest edge is contracted, fusing two adjacent components to form a larger-sized component.

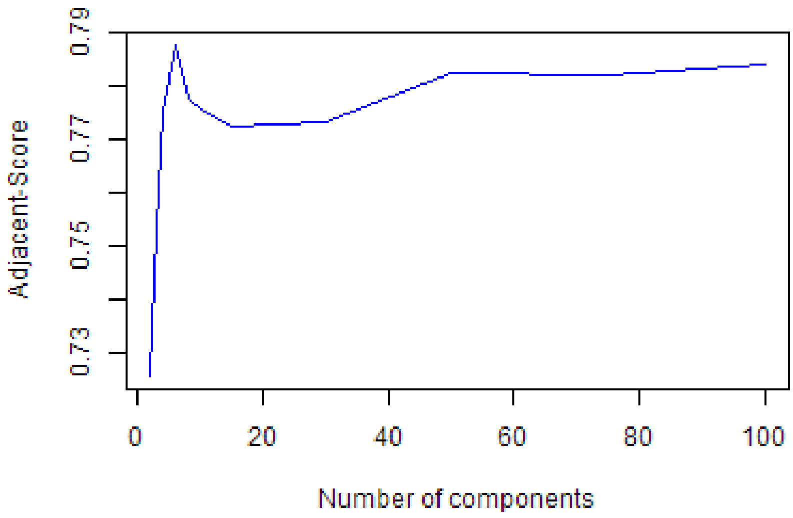
3. For all components adjacent to the fused component, all edges connecting them to the fused component are averaged into a single edge.

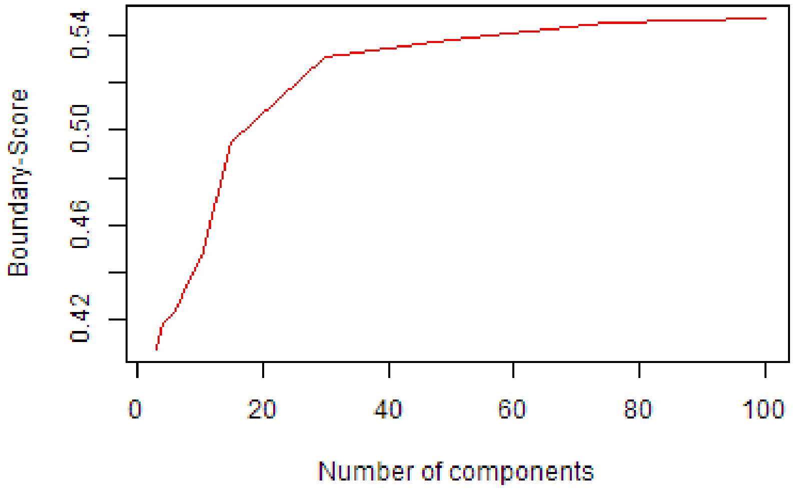
In short, the Edge-Contraction algorithm is a greedy heuristic that attempts to achieve the two objectives of balancing component size and maximizing the average edge weight within each component.

4.3.1 Implementation using Augmented Adjacency List

4.3.2 Optimal Number of Components

The Edge-Contraction algorithm has the benefit of requiring only one parameter, the target component number. We assessed the validity of EC parcellations with varying numbers of components. The plots below show how the Adjacent-Score and Boundary-Score varied according to the number of components in the EC algorithm. The peak in the Adjacent-Score occurred at 6 components.


Adjacent-Score
0.73
0.75
0.77
0.79
0
20
40
60
80
100
Number of components


Boundary-Score
0.42
0.46
0.50
0.54
0
20
40
60
80
100
Number of components

Chapter 5

Spectral Methods

In the previous chapter, we showed how local search heuristics produced parcels that were balanced and had high within-parcel and low between-parcel edge weights. The central idea behind such methods was to choose vertices to be in the same component if the edge connecting them has high distance correlation. Vertices were added to components one-by-one with constraints on component size, but not on component shape. As a result, one salient issue with these parcellations was lack of smoothness, or regularity in the parcels' spatial shapes. There was scant resemblence between the anatomical maps of the brain depicting smooth, rotund lobes and our jagged, web-like parcellations.

One key reason for this phenomenon are the local search heuristics' focus on maximizing *average* within-component edge weights (equivalently, minimizing *average* between-component edge weights because edges are either within the same component or between different components). To get smoothness in the boundary between components, we could either impose a penalty for too many between-component edges and work that into the local search heuristics, or try minimizing over the sum of all weights on between-component edges. This chapter deals with the second approach and this family of methods is called spectral partitioning.

Spectral partitioning constitutes the second major class of techniques used to partition graphs. Rather than rely on local component-growing heuristics, spectral partitioning uses information about the entire graph at once.

Throughout this chapter, a valid partitioning $P_k = (V_1, ..., V_k)$ of the graph $G = (V, E)$ is defined in the same way as in chapter 3; i.e., it must satisfy

1. $V_i \neq \varnothing$ for all $V_i \in \mathcal{P}_k$
2. $\bigcup\limits_{i=1}^{k} V_i = V$
3. $V_i \cap V_j = \varnothing$ for all $V_i, V_j \in \mathcal{P}_k$
4. V_i is connected (i.e. for every two vertices in V_i, there is a path between them) for all $V_i \in \mathcal{P}_k$

For all edges $(i,j) \in E$, let w_{ij} denote the weight of the edge connecting vertices i and j. $S^{n \times n}$ is the set of real symmetric $n \times n$ matrices. We further define, for a given graph $G = (V, E)$, the associated

Definition 5.0.1 *Adjacency matrix. $A \in \mathcal{S}^{n \times n}$ has entries*

$$A_{ij} = \begin{cases} w_{ij} & \textit{if } (i,j) \in E \\ 0 & \textit{otherwise} \end{cases}$$

Definition 5.0.2 *Degree matrix. $D \in \mathcal{S}^{n \times n}$*

$$D_{ij} = \begin{cases} \sum_{k=1}^{n} A_{ik} & \textit{if } i = j \\ 0 & \textit{otherwise} \end{cases}$$

5.1 Size-Constrained MinCut and Graph Bipartitioning

Consider the case $k = 2$. For all $i \in V$, let $x_i = 1$ if $i \in V_1$ and $x_1 = -1$ if $i \in V_2$. Then the sum of weights on edges between the two components is

$$\begin{aligned} C(P_2) &= \sum_{i \in V_1} \sum_{j \in V_2} A_{ij} \\ &= \sum_{i=2}^{n} \sum_{j=1}^{i-1} \frac{(x_i - x_j)^2}{4} A_{ij} \end{aligned}$$

since

$$(x_i - x_j)^2 = \begin{cases} 4 & \text{if } i, j \text{ are in different components} \\ 0 & \text{otherwise} \end{cases}$$

$C(P_2)$ can also be written in a matrix quadratic form, as

$$
\begin{aligned}
C(P_2) &= \sum_{i=2}^{n} \sum_{j=1}^{i-1} \frac{(x_i - x_j)^2}{4} A_{ij} \\
&= \frac{1}{2} \sum_{i,j=1}^{n} \frac{(x_i - x_j)^2}{4} A_{ij} \\
&= \frac{1}{2} \sum_{i,j=1}^{n} \frac{x_i^2 + x_j^2 - 2x_i x_j}{4} A_{ij} \\
&= \frac{1}{2} \sum_{i,j=1}^{n} \frac{1 - x_i x_j}{2} A_{ij} \\
&= \frac{1}{4} \sum_{i,j=1}^{n} (x_i^2 - x_i x_j) A_{ij} \\
&= \frac{1}{4} \sum_{i=1}^{n} x_i^2 \sum_{j=1}^{n} A_{ij} - \frac{1}{4} \sum_{i,j=1}^{n} x_i A_{ij} x_j \\
&= \frac{1}{4} \sum_{i=1}^{n} x_i^2 D_{ii} - \frac{1}{4} x^T A x \\
&= \frac{1}{4} x^T (D - A) x \\
&= \frac{1}{4} x^T L x
\end{aligned}
$$

where L is called the Laplacian matrix of the graph and defined as $L = D - A$. MinCut can thus be formulated as minimizing $x^T L x$ subject to $x \in \{-1, 1\}^n$.

Algorithms like Karger's can solve MinCut in polynomial time. However, MinCut in this formulation lacks constraints on the size of the partitions, and if applied to our brain parcellation problem, would result in severely inbalanced partitions. If constraints on the sizes of the components were added, the problem becomes NP-hard [citation].

An old but effective approach to bipartitioning uses the eigenvectors of the Laplacian matrix and is called spectral bipartitioning. The approach relaxes the $\{-1, 1\}$ constraint on x (and rescales x) so that it need only satisfy $\|x\| = 1$ ($\|\cdot\|$ here refering to L2 norm). It is easy to see that $\big\{x : x \in \{-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\}^n\big\} \subset \big\{x \in \mathbb{R}^n : \|x\| = 1\big\}$ The problem now becomes

$$
\begin{aligned}
\min_{x} \quad & x^T L x \\
\text{s.t.} \quad & \|x\| = 1
\end{aligned}
\tag{5.1}
$$

Using Lagrangian multipliers, it can be shown that all optimal solutions to the above must satisfy $Lx = \lambda x$ and this problem reduces to finding the smallest eigenvalues of L and their associated eigenvectors. In addition, 5.1.1 below implies that all eigenvalues are nonnegative.

Theorem 5.1.1 *Let L be a Laplacian matrix. Then $L \succeq 0$ (L is positive semidefinite)*

Proof. Let $x \in \mathbb{R}^n$. $x^T Lx = x^T Dx-$

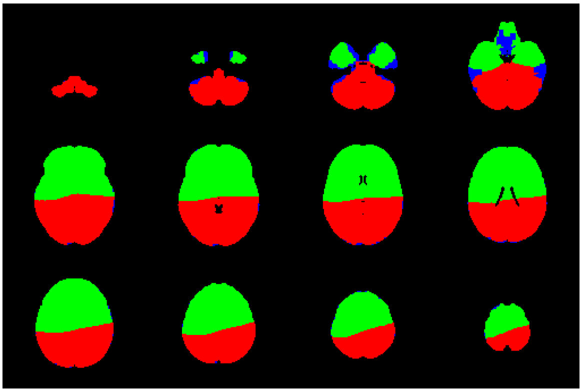
Note that from the $C(P_2) = \sum_{i>j} \frac{(x_i - x_j)^2}{4} A_{ij} = \frac{1}{4} x^T Lx$ equivalence we know that 0 and $(\frac{1}{\sqrt{n}}, ..., \frac{1}{\sqrt{n}})^T$ is a minimum eigenvalue and eigenvector to this system. For bipartitioning, the useful eigenvector is the one that corresponds to the 2nd smallest eigenvalue, which is nonzero if the graph as a whole is connected. We'll denote this eigenvalue as λ_1 and corresponding unit eigenvector as x_1. We have the following:

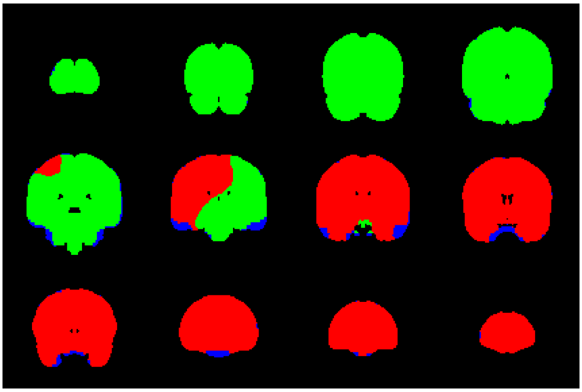
Theorem 5.1.2 *Let P_2 be any valid partition into 2 components. Then $C(P_2) \geq \lambda_1$*

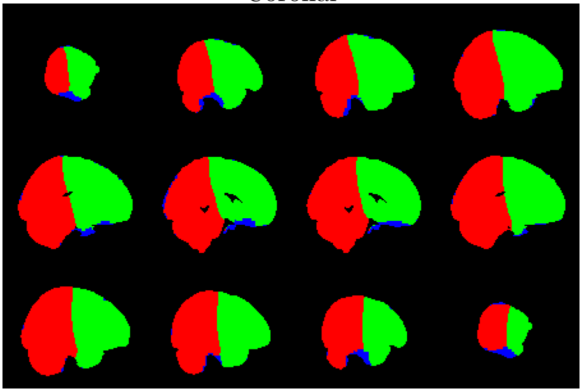
Proof.

In the literature, x_1 is often refered to as the Fiedler vector, after the first mathematician who studied it in detail [Fiedler 1975]. From the Fiedler vector we can obtain a variety of "good" bipartitions. We can impose a size constraint $|V_1| = s$ and obtain a bipartition satisfying this by placing the vertices associated with the s largest entries of x_1 in V_1. This encompasses bipartitions of equal component size. We can also sort the entries of x_1 and find the largest difference between consecutive sorted entries. Vertices corresponding to entries sorted to the left of this split can be placed in V_1 and vertices sorted to the right in V_2. This method tends to approximate the MinCut solution.

The result of spectral bipartitioning on a resting state fMRI scan is shown below. As anticipated, the boundaries of between the components are smooth.


Axial


Coronal


Sagittal

One can recursively apply this bipartitioning method to the component subgraphs to obtain k-partitions, but there is a more elegant approach involving additional eigenvectors that requires the construction of only one Laplacian matrix, which we shall discuss next.

5.2 Spectral k-partitioning

We'll begin with two definitions to set up the machinery for partitioning into k components.

Definition 5.2.1 *Assignment matrix.* $X \in \{0,1\}^{n \times k}$ *has entries*

$$X_{ih} = \begin{cases} 1 & \textit{if vertex } i \in V_h \\ 0 & \textit{otherwise} \end{cases}$$

Let u_m denote a vector of m ones. An assignment matrix characterizes a valid partition only if it satisfies $Xu_k = u_n$ and $X^T u_n > 0$. The columns of X are orthogonal.

Definition 5.2.2 *Partition matrix.* $P \in \{0,1\}^{n \times n}$ *has entries*

$$P_{ij} = \begin{cases} 1 & \textit{if vertices } i \textit{ and } j \textit{ are in the same component} \\ 0 & \textit{otherwise} \end{cases}$$

If P and X refer to the same partitioning, then $P = XX^T$.

In the k-component case, we define the weight of a partition $C(P_k)$ as the sum of weights of edges between different components (between-edges). This is equivalent to the definition below:

Definition 5.2.3 (Cut weight) *For a partition* $P_k = (V_1, ..., V_k)$*, the cut weight is defined as*

$$C(P_k) = \sum_{h=1}^{k} E_h$$

where E_h *is the sum of the weights of all edges with one vertex in* V_h *and one vertex not in it.*

This is equal to the sum of the weights of all edges in the graph minus the sum of the weights of all edges connecting vertices in the same component (within-edges).

In addition, if D is the degree matrix, then

$$\begin{aligned} \mathrm{Tr}(PD) &= \sum_{i,j=1}^{n} P_{ij} D_{ij} \\ &= \sum_{i=1}^{n} P_{ii} D_{ii} \\ &= \sum_{i=1}^{n} P_{ii} \sum_{j=1}^{n} A_{ij} \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} \end{aligned}$$

is the sum of the weights of all edges in the graph. Similarly, $\mathrm{Tr}(PA)$ equals the sum of weights of all within-edges. It follows that

$$C(P_k) = \mathrm{Tr}(X^T L X)$$

The derivation of the spectral relaxation for the multi-partition problem in [Chan 1994] actually doesn't attempt to minimize $C(P_k)$ directly for reasons to be elaborated below. Rather, it minimizes a related objective called the ratio-cut cost.

Definition 5.2.4 (Ratio-cut cost) *For a given partition* $P_k = (V_1, ..., V_k)$ *the ratio-cut cost* C_R *is defined*

$$C_R(P_k) = \sum_{h=1}^{k} \frac{E_h}{|V_h|}$$

where E_h is the sum of the weights of all edges with one vertex in V_h and one vertex not in it.

[Chan 1994] defines a new decision variable R, of the same form as X but with columns rescaled so that the column sum is $\sqrt{|V_h|}$ for each component V_h.

Definition 5.2.5 (Ratioed Assignment Matrix) *$R \in \mathbb{R}^{n\times k}$ has entries*

$$R_{ih} = \begin{cases} \frac{1}{\sqrt{|V_h|}} & \textit{if vertex } i \in V_h \\ 0 & \textit{otherwise} \end{cases}$$

The ratioed assignment matrix relates to the ratio-cut cost in the same way the assignment matrix relates to cut weight; namely, if R characterizes a partition P_k then

$$C_R(P_k) = \mathrm{Tr}(R^T L R)$$

R has the additional useful property that $R^T R = I$. This property leads to a closed form optimal solution to

$$\begin{aligned} \min_{R\in\mathbb{R}^{n\times k}} \quad & \mathrm{Tr}(R^T L R) \\ \text{s.t.} \quad & R^T R = I \end{aligned} \tag{5.2}$$

[Fan 1949] proved that an optimal solution $\hat{R}$ to the above consists of k orthonormal eigenvectors corresponding to the k smallest eigenvalues of L. Analogously to $P = XX^T$, $\hat{R}$ can be thought of as n k-dimensional points where the dot products of the ith and jth rows measure the affinity of vertices i and j to be in the same component.

As [Chan 1994] pointed out, clustering would be more accurate if it is performed on the un-ratioed assignment matrix, since the ratioed assignment matrix R has the problematic property that for any i, j in the same component, the dot product $R_i^T R_j$ depends on the size of that component. The un-ratioed version of $\hat{R}$ be recovered by dividing each row by its Euclidean norm. The result, $\hat{X}$, can be thought of as n points in $\mathbb{R}^k$ embedded on the surface of the unit hypersphere.

Obtaining the partition assignment matrix X, from this spherical embedding is a clustering problem. We used k-means with cosine similarity $s(x, y) = x^T y$ for this purpose. For each cluster h and its associated points matrix $H \in \mathbb{R}^{m\times k}$, the location of the cluster centroid c_h in the next iteration satisfies $\sum_{i=1}^{m} H_i = \lambda c_h$ for some scalar λ and $\|c_h\| = 1$.

Chapter 6

Linear and Semidefinite Optimization