Brain Functional Parcellation with Energy Distance Correlation

Felix Xiao

January 31, 2016

Contents

Chapter 1

Functional MRI Data and Brain Parcellation

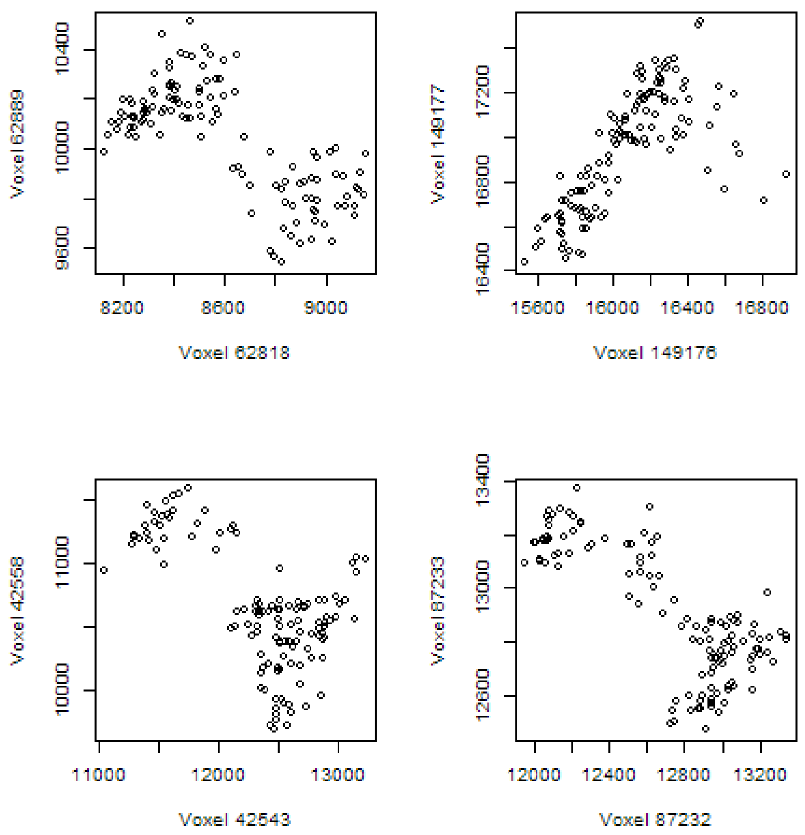
[Need to find somewhere to put this] I conducted all parcellation and validation procedures on the ABIDE 50002 fMRI data set provided by Kevin Lin. This data set contains 233305 voxels and 124 time samples. Spatial information is encoded as a graph; each voxel is represented by a vertex, and each vertex 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).

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 s̈imilar.̈ 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.

[insert figure here]

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 [fact check needed here]. As a result in many of the edges between different parcels, the corresponding voxels would be strongly dependent with negative correlation [fact check needed here].

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

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

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*

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

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

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

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
$$\mathcal{V}(v_1 + a_1Q_1X, v_2 + a_2Q_2Y) = \sqrt{|a_1a_2|}\mathcal{V}(X,Y)$$

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 k-fold partition of the graph is a collection of k disjoint vertex subsets (called parcels) $P_1, ..., P_k \subset V$ satisfying $P_1 \cup ... \cup P_k = V$. We impose the additional requirement that each parcel is a connected component; i.e. that there exists a path between any pair of vertices in a parcel. For this reason we'll use the terms parcel and component interchangeably.

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

Definition 3.2.1 *Between-Score.*

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

For two different parcels P and P', let $E_{P,P'}=\{(i,j)\in E : i\in P \text{ and } j\in P'\}$

Definition 3.2.2 *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 s̈earchf̈or 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:

4.3 Edge Contraction

The Edge-Contraction algorithm attempts to address the problem of poor Adjacent-Score of the Add-Edge parcellations relative to the same algorithm applied to the randomized graph. 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 ädded·. 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

- the smallest component with the largest edge to another component is selected - this largest edge is contracted, fusing two adjacent components to form a larger-sized component - for all components adjacent to the fused component, all edges connecting them to the fused component are averaged into a single edge

The Edge-Contraction algorithm is thus 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

[...]

Optimal Number of Components

After running the Edge-Contraction algorithm for a variety of different target components, we arrived at the following result:

Chapter 5

Optimization Approaches

In this chapter we introduce a different approach to the graph partitioning problem. Rather than use local search heuristics to find candidate partitions and evaluate them using pre-defined criteria, the optimization approach uses or adapts one or more of the validation criteria in chapter 2 as an objective function and seeks to maximize or minimize this objective using the vertex partition assignments as decision variables.

We will start with a simple idea to frame the brain parcellation task as an optimization problem. We will then repeatedly refine the optimization problem to let it satisfy our notions of what constitutes a good parcellation and be computationally tractable.

For all edges $(i, j) \in E$, let w_{ij} denote the weight of the edge connecting vertices i and j.

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

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

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

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

5.1 Edge Weight Minimizing Bipartition

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. However, one salient issue with these parcellations was lack of smoothness, or regularity in the parcels' spatial shapes.

An effective way of encouraging parcel smoothness is to minimize the sum of weights on all edges connecting a parcel to a different parcel. This objective has the simultaneous effect of encouraging sharp boundaries between adjacent parcels.

Consider the case of two parcels, labeled -1 and 1. Let $x \in \{-1, 1\}^n$ and for each vertex $i = 1, ..., n$, x_i indicate the parcel to which it belongs. The sum of edge weights on the cut, or boundary between the two parcels is $\sum_{(i,j)\in E}(x_i - x_j)^2 w_{ij}$ where $(x_i - x_j)^2 = 1$ if and only if vertices i and j are in different parcels. Further

$$\begin{aligned}\sum_{(i,j)\in E}(x_i-x_j)^2w_{ij} &= \sum_{(i,j)\in E}(2x_i^2-2x_ix_j)w_{ij}\\ &= \sum_{i,j=1}^{n}(x_i^2-x_ix_j)A_{ij}\\ &= \sum_{i=1}^{n}x_i^2\sum_{j=1}^{n}A_{ij}-x^TAx\\ &= \sum_{i=1}^{n}x_i^2D_{ii}-x^TAx\\ &= x^T(D-A)x\\ &= x^TLx\end{aligned}$$

where L is called the Laplacian matrix of the graph and defined as $L = D - A$. Hence the problem of finding the minimum cut of a graph can be reduced to solving the combinatorial optimization problem

$$\begin{aligned}&\min_x x^TLx\\ &\text{s.t.} x\in\{-1,1\}^n\end{aligned} \tag{5.1}$$

Algorithms like Karger's can solve the Min Cut problem in polynomial time. However, the Min Cut problem in this formulation lacks constraints on the size of the partitions. If applied to our brain parcellation problem, the result would likely be one miniscule parcel with just one or two voxels and one enormous parcel constituting the entire remainder of the brain.

To address this issue, we must add an additional size constraint to 5.1. If we want one partition to have b more voxels than the other, then a simple re-formulation might be

$$\begin{aligned}&\min_x x^TLx\\ &\text{s.t.} x\in\{-1,1\}^n\\ &\sum_{i=1}^{n}x_i=b\end{aligned} \tag{5.2}$$

With this additional constraint, now the problem becomes NP-complete [fetch citation from Buluc 2013] Fortunately, a widely used relaxation of the first constraint to $\|x\|_2 = n$ converts the combinatorial optimization problem into a tractable continuous optimization problem, for which there exists an analytic solution derived using Lagrange multipliers.

5.2 Constrained k-Part Graph Partitioning