Higher Order Interactions

Moo K. Chung
University of Wisconsin-Madison
mkchung@wisc.edu

Abstract

In resting-state functional brain imaging (rs-fMRI), the investigation of higher-order interactions between brain networks is essential for understanding the complex organization of neural activity. We aim to explore whether such interactions can be consistently observed across different time points and subjects. To model higher-order interactions, we employ techniques from topological data analysis, specifically persistent homology, which captures the multi-scale connectivity patterns inherent in rs-fMRI data. This approach allows us to characterize not only pairwise connections between brain regions but also the presence of higher-order structures, such as loops and voids, that reveal deeper organizational principles of brain networks. Our findings suggest that certain topological features exhibit consistency across individuals and time, indicating the existence of robust higher-order network dynamics. We further discuss the implications of these results for understanding functional connectivity and propose persistent homology as a novel tool for investigating complex brain interactions beyond traditional graph-theoretic approaches.

1 Measuring Higher-Order Interactions

Given a multivariate time series $X(t) = [X_1(t), X_2(t), \dots, X_p(t)]$, where each $X_i(t)$ represents the activity of a node (e.g., a brain region in fMRI), we aim to identify higher-order interactions beyond pairwise connections. A simple method for detecting such interactions is based on persistent homology, which is a part of topological data analysis (TDA). To measure interactions, we first estimate the connectivity between the time series. A common approach is to compute a pairwise similarity measure such as

$$C_{ij} = \text{corr}(X_i(t), X_j(t))$$

where $\text{corr}(\cdot, \cdot)$ is the Pearson correlation coefficient. This results in a weighted connectivity matrix C, where each entry represents the strength of interaction between nodes.

Instead of analyzing only pairwise interactions, we extend to higher-order interactions by constructing a simplicial complex. Given the weighted network

C, we define a simplicial complex, where nodes (0-simplices) represent individual time series, edges (1-simplices) are formed if $C_{ij} > \tau$, where τ is a threshold. Triangles (2-simplices) are included if all three edges among three nodes exist and satisfy $C_{ij}, C_{ik}, C_{jk} > \tau$. The thresholds τ is usually determined statistically. Higher-order simplices are formed similarly.

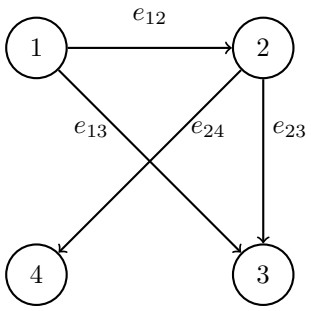
2 Construction of the Boundary Matrices

While the simplicial complex provides a structural representation by including nodes, edges, triangles, and higher-order simplices based on connectivity thresholds, it does not inherently quantify the significance of these interactions. The transition from simplicial complexes to boundary matrices is crucial for quantifying higher-order interactions in multivariate time series. To systematically measure higher-order interactions, we construct boundary matrices ∂_k, which encode the relationships between k-simplices and their $(k-1)$-dimensional faces. These matrices allow us to compute homology groups that reveal topological features such as loops (1-cycles) and voids (2-cycles), which go beyond pairwise interactions. *The presence of a triangle in the simplicial complex does not necessarily indicate a higher-order interaction unless it forms a persistent topological feature, such as a loop that remains across multiple filtration scales.* Persistent homology provides a framework to track the birth and death of these features by analyzing the null spaces and ranks of the boundary matrices, leading to the computation of Betti numbers that quantify the number of independent topological structures. Thus, the construction of boundary matrices is essential for extracting topological invariants from the simplicial complex, enabling a rigorous and scalable method to detect stable higher-order interactions in complex networks.

Each boundary matrix ∂_k encodes the relationships between k-simplices and their $(k-1)$-dimensional faces. The boundary matrices are constructed iteratively as follows. The 1-boundary matrix ∂_1 describes the relationship between edges (1-simplices) and nodes (0-simplices). Each edge e_{ij} connects two nodes v_i and v_j, and the **boundary operator** is defined as:

$$\partial_1(e_{ij}) = v_j - v_i$$

For instance, the **boundary matrix** corresponding to the boundary operator for the following directed graph



is given by

$$\partial_1 = \begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}.$$

Each row corresponds to an edge, and each column corresponds to a node. The graph is drawn using `tikzpicture` package in LaTex, which has limited functionality.

The 2-boundary matrix ∂_2 describes the relationship between triangles (2-simplices) and edges (1-simplices). Each triangle t_{ijk} consists of three edges e_{ij}, e_{ik}, e_{jk}, and its boundary is given by

$$\partial_2(t_{ijk}) = e_{ij} + e_{jk} - e_{ik}.$$

For instance, in matrix form, the boundary matrix ∂_2 can be given

$$\partial_2 = \begin{bmatrix} 1 & -1 & 1 & 0 \\ 0 & 1 & -1 & 1 \end{bmatrix}.$$

Each row represents a triangle, and each column represents an edge. Unfortunately, drawing a simplicial complex corresponding to a complex boundary matrix with the `tikzpicture` package is not feasible. Therefore, we implemented the iterative procedure in MATLAB within the PH-STAT package.

2.1 3-Boundary Matrix ∂_3 (Tetrahedra to Triangles)

The 3-boundary matrix ∂_3 encodes the relationship between tetrahedra (3-simplices) and their boundary triangles (2-simplices). A tetrahedron τ_{ijkl} is formed by four triangular faces $t_{ijk}, t_{ijl}, t_{ikl}, t_{jkl}$, and its boundary is given by

$$\partial_3(\tau_{ijkl}) = t_{ijk} - t_{ijl} + t_{ikl} - t_{jkl}$$

In matrix form, it can be represented as

$$\partial_3 = \begin{bmatrix} 1 & -1 & 1 & -1 \end{bmatrix}.$$

This example only shows a single tetraherdon but in a more complex simplicial complex, each row represents a tetrahedron, and each column represents a triangle in ∂_3.

Exercise. Note the graph visualization using `tikzpicture` for ∂_2 is done using CHAT-GPT. Unfortunately, CHAT-GPT cannot draw complex graph structue for ∂_3. Given a boundary matrix ∂_3, write a Matlab function that draws graph structure in 2D or 3D. You need to assume nodes positions are given along a circle or a sphere.

2.2 General Formula for Boundary Matrices

For any k-simplex $\sigma^k = [v_0, v_1, \ldots, v_k]$, its boundary consists of $(k-1)$-dimensional faces. The boundary matrix ∂_k is computed iteratively using

$$\partial_k(\sigma^k) = \sum_{i=0}^{k} (-1)^i [v_0, v_1, \ldots, \hat{v}_i, \ldots, v_k],$$

where $\hat{v}_i$ denotes that the vertex v_i is removed. The matrix representation is then given by

$$(\partial_k)_{ij} = \begin{cases} +1, & \text{if } \sigma_j^{k-1} \text{ is a face of } \sigma_i^k \text{ with aligned orientation,} \\ -1, & \text{if } \sigma_j^{k-1} \text{ is a face of } \sigma_i^k \text{ with opposite orientation,} \\ 0, & \text{otherwise.} \end{cases}$$

This iterative process constructs the higher-dimensional boundary matrices from lower-dimensional ones.

The boundary matrices $\partial_1, \partial_2, \ldots$ are computed iteratively to track the emergence and disappearance of topological features such as loops (1-cycles) and voids (2-cycles). Using persistent homology, we analyze the lifespan of these features in a **persistence diagram**, where long-lived features indicate significant higher-order interactions.

3 Computational Complexity

A brute-force approach that computes each ∂_k independently can be inefficient. Instead, one can build ∂_{k+1} iteratively from ∂_k. For example, going from ∂_1 (edges) to ∂_2 (triangles) requires finding all node triplets that form triangles. In a dense graph with p nodes and $\mathcal{O}(p^2)$ edges, there can be $\mathcal{O}(p^3)$ such triplets, making the naive enumeration $\mathcal{O}(p^3)$. However, for sparse graphs or simplicial complexes where $E \ll p^2$, more efficient methods such as intersecting neighbor sets can achieve around $\mathcal{O}(E^{3/2})$ [Schank and Wagner(2005)]. Once a triangle is identified, populating the corresponding row in ∂_2 is straightforward and adds only linear overhead in the number of triangles.

When constructing ∂_{k+1} from ∂_k, a similar principle applies for higher dimensions. Suppose each k-simplex contains nodes that each have at most k neighbors *within* that simplicial structure. In other words, for each node of a given k-simplex, there are at most k other nodes that share a simplex with it. Under this assumption, intersecting the neighbor sets of k nodes takes $\mathcal{O}(k)$ time. If n_k is the number of k-simplices, the total run time to identify all $(k+1)$-simplices is thus $\mathcal{O}(n_k \cdot k)$. This is much smaller than the worst-case $\mathcal{O}(p^k)$, and in practice remains feasible for moderate k, especially in sparse rs-fMRI networks with constrained node degrees.

Here, we give a more detailed explanation on the **Intersecting Neighbor Sets** method. Suppose we have an undirected (or bidirectional) graph $G = (V, E)$ and we want to find all triangles. The key observation is that a triangle (u, v, w) exists precisely when each pair among $\{u, v, w\}$ is connected by an edge. In the neighbor-set approach, we iterate over each edge (u, v) and look for common neighbors of u and v. Any node w that is a neighbor of both u and v completes a triangle (u, v, w).

$$\text{Triangles} = \big\{(u, v, w) \;\big|\; w \in \Gamma(u) \cap \Gamma(v)\big\},$$

where $\Gamma(x)$ denotes the neighbor set of node x. By using adjacency lists rather than an adjacency matrix, intersecting neighbor sets can be done more quickly than a naive $\mathcal{O}(p^3)$ enumeration for dense graphs. For instance, if each node has $\deg(u)$ neighbors, the intersection step $\Gamma(u) \cap \Gamma(v)$ can be performed in $\mathcal{O}(\min\{\deg(u), \deg(v)\})$ time by iterating over the smaller neighbor set and checking membership in the larger set. This yields lower complexity in many practical scenarios where graphs are not completely dense. As soon as we identify a node w in the intersection, we know (u, v, w) forms a triangle. Once the triangles are discovered, constructing the corresponding rows in the ∂_2 boundary matrix becomes straightforward: each triangle (u, v, w) corresponds to a row, and its edges (u, v), (v, w), and (u, w) determine the columns where ± 1 entries are placed.

The Matlab function

```
function S = PH_connectivity2simplex(C, tau, k)
```

inputs the correlation matrix `C` and threshold `tau` and the maximum dimension of the simplical complex `k`. For instance, to build 4-nodes connectivity (tetrahedra in 3D), `k` should be 3. Then the function outputs the simplicial complex `S`.

4 Checking for Higher-Order Interactions

To assess whether a network exhibits higher-order interactions, we compute its Betti numbers, which quantify topological features beyond simple pairwise connections. Betti numbers are derived from the network's boundary matrices,

which encode the incidence relationships between simplices of different dimensions. Specifically, the k-th Betti number, β_k, is computed as

$$\beta_k = \dim(\ker \partial_k) - \operatorname{rank}(\partial_{k+1}),$$

where ∂_k is the boundary matrix that maps $(k-1)$-simplices to k-simplices. In practice, we focus on identifying the highest dimension k for which $\beta_k > 0$. The presence of any nonzero Betti number beyond the zeroth level indicates that the network contains complex structures—such as loops (β_1) or voids (β_2)—which are indicative of higher-order interactions.

References

[Schank and Wagner(2005)] T. Schank and D. Wagner. Finding, counting and listing all triangles in large graphs, an experimental study. In *International workshop on experimental and efficient algorithms, LNCS*, volume 3503, pages 606–609. Springer, 2005.