

APPROXIMATING CUTNORM: A ROBUST METHOD TO COMPUTE DISTANCE
BETWEEN DENSE GRAPHS FOR PREDICTION AND INTERPRETATION

BY

PING-KO CHIU

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Adviser:

Assistant Professor Oluwasanmi Koyejo

ABSTRACT

This thesis presents techniques of modeling large and dense networks and methods of computing distances between them. Large and dense networks arise in many disciplines. Through recent advancements in dense graph theory and graph convergence, we have a new perspective on how large graphs should be considered and how the similarity of graphs should be computed. The thesis discusses the steps to approximate the distance between graphs and the integration of a new search algorithm to accelerate computation. A software package is produced to estimate distances between graphs and made available as the *Cutnorm* package on PyPI. The algorithm and software shows great performance on theoretical models and is faster than existing implementations. The thesis also explores practical applications of the graph convergence theory and Cut-Distances. It presents the theory and techniques to analyze human brain connectivity graphs from the *ADHD200* dataset of the *1000 Connectome Project*. It also presents a new insight to monitoring Artificial Neural Network convergence during the training process.

ACKNOWLEDGMENTS

I thank my adviser, Professor Oluwasanmi Koyejo, for his guidance and knowledge. He cares deeply for his students and provides us the resources to succeed. It was a pleasure to be taught by him and to work along side him. I also thank Dr. Peter Diao for the inspiration and collaboration on the Cutnorm Project. His ideas and excitement for the project keep us focused. I would also like to thank Alan Wang for his collaboration on the work of neural network representation learning.

I am grateful to my mother and father, Hannah Chen and Yueh-Chin Chiu, and my sister, Holly Chiu, for their endless love and support. Together, we make the most efficient and high functioning team.

I am thankful to Cleona Tsang, best friend and supporter. She kept me in check during times of optimism and gave me support in difficult times.

I thank my dear friends from childhood and from UIUC. They cheered me on along the way and created home away from home.

My gratitude goes towards all Professors, instructors, and staff at UIUC that I have interacted with. It is a privilege to be a part of such a great institution.

TABLE OF CONTENTS

| | |
|---|----|
| CHAPTER 1: INTRODUCTION | 1 |
| CHAPTER 2: DISTANCE BETWEEN GRAPHS | 3 |
| CHAPTER 3: CUTNORM APPROXIMATION | 10 |
| CHAPTER 4: CUTNORM PACKAGE..... | 18 |
| CHAPTER 5: APPLICATIONS AND RESULTS | 22 |
| CHAPTER 6: DISCUSSION AND CONCLUSION..... | 39 |
| REFERENCES | 41 |

CHAPTER 1: INTRODUCTION

Large graphs arise in many problems and disciplines. Graphs model relationships between entities (nodes). Thus many of the structures and problems in the world can be modeled as graphs. Graphs with large number of nodes are large graphs. An example of a large graph is the human brain. It has billions of neurons and each of them can be represented as nodes and their connectivity as edges. The internet has billions of endpoints that are connected and transmitting information. Each of the endpoints can be represented as nodes and the information transmitted as edges. Statistical physics model interactions between large numbers of particles. The particles can be represented as nodes and their interactions as edges.

These problems call for specialized mathematics and computational techniques. Many really large graphs cannot be computationally expressed as a whole and only samples of the large graphs can be considered. Thus special care needs to be taken to evaluate properties about them.

This thesis explores the distances between graphs. While much work has been done on the theory of properties within a single graph such as the distances between nodes, properties of graphs within a family of graphs are of interest to many applications.

One of such application is the human connectome, a map of the neuron connections in the human brain. Given the human brain connectivity graph between a diseased patient and a healthy control, what is the distance between the them? Is it possible to amplify distances between differing graphs for predictive purposes? How can we approximate this distance? The sheer number of neurons in the brain makes this a very large graph problem.

To answer these questions, we need to study the convergence of graphs, infinite graphs, graph limit objects, measure preserving maps, notion of distance with respect

the graphs, and the methods of approximating the distance.

The thesis introduces a new search algorithm for approximating the Cut-Distance between two graphs. We also produced open source software for general adoption of the technique.

My work relies heavily on the works of Lovasz and Borgs on the theory of dense graph convergence [1], [2], the works of Alon, Noar, Freize, and Kannan on Cutnorm and approximation techniques [3], [4], and the works of Wen and Yin on optimization algorithms [5].

Most of Chapter 2 and 3 will summarize previous work and provide more detailed explanations and analysis to certain concepts to set up for the techniques and applications in later chapters.

CHAPTER 2: DISTANCE BETWEEN GRAPHS

2.1 Convergence of Graphs

A simple finite graph $G = (V, E)$ contains vertex set $V = \{v_1, \dots, v_n\}$ and edge set $E \subseteq V \times V$. Let the adjacency matrix $A = a_{i,j}$ where $a_{i,j} = 1$ if $(v_i, v_j) \in E$ and $a_{i,j} = 0$ otherwise. A is a representation of the connectivity of the graph. The definition can be generalized to incorporate weighted edges and weighted vertices.

We can trivially incorporate weighted edges by replacing the 1 in the adjacency matrix with the edge weights and incorporate weighted nodes by defining $\alpha(v)$ as weights of each vertex where $\sum_{v \in V} \alpha(v) = 1$.

Homomorphisms between two graphs G and H are mappings $f : V(G) \rightarrow V(H)$ such that that $(f(u), f(v)) \in E(H)$ for every $(u, v) \in E(G)$. Alternatively, homomorphisms preserve adjacency between graphs. The notion of homomorphism is helpful when considering the global structural similarity between different graphs of varying sizes.

Many graph problems we observe in nature are graphs as a sample of larger graphs. For an example, the graph of the internet is large but only a portion of it can be sampled at a time. When we know precisely that the graphs we are studying are samples of larger graphs, it is especially important that we assess their properties with the larger graphs in consideration.

So far, we have only considered finite graphs and standard definitions for them. Large and infinite graphs are important as we generalize the sampling of small graphs from large or infinite graphs.

Several ground-breaking theories have been developed on the convergence of dense graphs in recent years. Lovasz and Szegedy [6] has generalized the notion of graph limits

to measurable symmetric functions in $\mathcal{W} : [0, 1]^2 \rightarrow [0, 1]$ referred to as graphons. The measurable symmetric functions are representations of infinite graphs. Consider the adjacency matrix of sequence of graphs G_n as $n \rightarrow \infty$ where n represents the number of vertices of the graph. Graphons are the limit objects for which a graph sequence G_n converges to. Borgs et al. [2] has defined the convergence sequences of graphs in terms of homomorphism densities from small graphs to large graphs (left convergence) and from large graphs to small graphs (right convergence). Homomorphism from small graphs to large graph G is of particular interest to us due to its relation to the sampling of G and the distance between two small graphs. We can consider small graphs to be homomorphic to large graphs if there exists adjacency preserving mapping from the small graph to the large graph. A homomorphic sampling of the large graph G is then a small graph where the global structure is preserved.

The idea of sampling can be further extended as we consider infinite graphs in the space of \mathcal{W} and step functions $W \in \mathcal{W}$ as detailed by Lovasz [7]. Here we summarize Lovasz's findings to aid the discussion of the significance of the Cutnorm introduced later. Step functions W are graphons where $[0, 1]$ is partitioned into k distinct subsets $\mathcal{S} = \{S_1, \dots, S_k\}$ where each region in $\mathcal{S} \times \mathcal{S}$ is constant. Consider a finite graph G_n as $n \rightarrow \infty$, step functions W represent graphon limit objects for finite graphs. Each region in $\mathcal{S} \times \mathcal{S}$ represents some edge weight between the two sets of vertices.

Two graphons W and W' are weakly isomorphic if there exists another graphon U such that there exists measure preserving maps $\phi, \phi' : [0, 1] \rightarrow [0, 1]$ where $W(x, y) = U(\phi(x), \phi(y))$ and $W'(x, y) = U(\phi'(x), \phi'(y))$ for almost all $x, y \in [0, 1]$. The idea that graph sequences G_n converges to some limit object $W \in \mathcal{W}$ and that these limit objects can have weak isomorphisms between each other helps us define distances between graphs. Two graphs G and G' sampled from the same graphon may not be isomorphic or homomorphic, but the graphon that they converge to may be isomorphic or weakly

isomorphic. By looking at how weakly isomorphic two graphons are, we can then define distances between graphs.

2.2 Measure Preserving Maps

In the previous section, we introduced measure preserving maps $\phi, \phi' : [0, 1] \rightarrow [0, 1]$ that defines weak isomorphism between graphons. The discrete analogue would be a mapping of vertices, $\psi : V(G) \rightarrow V(H)$, between two graphs G, H .

Exactly how to find the measure preserving maps depends on the information of the vertex sets. Two properties of the vertex sets give information to the difficulty of finding the measure preserving maps: the cardinalities and the labels. In particular, we need to answer the questions: Are the cardinalities of the two vertex sets $|V_G|$ and $|V_H|$ equivalent?, and are the two vertex sets labeled?

If the cardinalities of the vertex sets are equal and the vertex sets are labeled, then the bijection between the two vertex sets is the measure preserving map. This can be achieved in constant time.

If the cardinalities of the vertex sets are not equal but the vertex sets are labeled, then the labels already give a surjection between the two vertex sets. The surjection can then be used to re-scale the graph so that there exists a bijection between the two new vertex sets. Additionally, if we have additional information regarding the problem, we can define measure preserving maps that are reasonable. For an example, we may wish to investigate the relationships of stock prices in the S&P 500 index over the past 40 years. However, the index list is dynamic; companies can be listed and unlisted from the index. If we wish to treat companies as individual entities, we may be limited to only monitoring companies that are on the list for the entirety of the date range that we are investigating. If we are interested in the relationships between various sectors, we can perform aggregation of the companies by their sector label.

The aggregation process is an example of a Measure Preserving Map. By mapping all companies within one sector into one statistic representing the sector, data of the 500 companies at different times can be properly compared with each other. Given labeled graphs that have different vertex set cardinalities, we can find measure preserving maps in polynomial time.

If the cardinalities of the vertex sets are equal but the vertex sets are not labeled, then a bipartite matching has to be done to find a bijection for the two vertex sets. The brute force method can be achieved in $O(n!)$ where n is the size of the vertex set.

If the cardinalities are not equal and the vertex sets are not labeled, then the entire function space needs to be searched over. This can be achieved in $O(|V(G)|^{|V(H)|})$.

From this point on in the thesis, we will assume that our problems are of the first two cases above: Either we already have a measure preserving map, or we can find a measure preserving map in polynomial time.

2.3 Using l_p Norm as Dissimilarity Measure

The vector l_p norm of a vector $\mathbf{x} = [x_1, \dots, x_n]$ is defined as

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^n |x_i|^p \right)^{1/p}$$

Given vectorized adjacency matrices A and B , the l_p norm of the difference is a dissimilarity measure. Let us define the distance between two graphs given by the l_p norm as

$$d_{l_p} := \frac{1}{n^{2/p}} \|(A - B)\|_p \tag{2.1}$$

The normalization enforces bounded solutions. It is determined according to the

supremum of each norm in the binary case, $\sup_{B \in \{0,1\}^{n^2}} \|B\|_p$. For an example, the l_p norms of the difference adjacency matrices of simple unweighted graphs is bounded by

$$0 \leq d_{l_p} \leq 1$$

Of the various l_p norms, the l_1 norm is the most straight forward and it represents the edit distance: For binary graphs, the number of edits needed to make one graph identical to the other. Given two identical graphs, the edit distance between the adjacency matrices is zero since every entry is identical. Given two binary graphs where each entry of one graph's matrix contains the binary complement of the other, the edit distance is one. The edit distance is the simplest notion of distance between the adjacency matrix representation of two graphs.

However, the l_p norms are inadequate for assessing global structural similarities between graphs. Lovasz discusses this through the example of Erdős-Rényi random graphs [1]. The simple Erdős-Rényi model $G(n, p)$ considers graphs of n vertices where each pair of vertices are connected independently with probability p [8]. The graphon limit of this object as $n \rightarrow \infty$ is $g : [0, 1]^2 \rightarrow p$.

Consider two independently generated Erdős-Rényi random graphs, A and B , of size n with probability 0.5. The l_1 norm of the difference matrix is 0.5 with high probability, but the graphon limit object of the two are identical. Thus, the l_1 norm of the difference matrix cannot represent structural similarity when considering the convergence of graphs to graphon limit objects. Since $l_1 \leq l_2 \leq l_\infty$, the other l_p norms do not adequately represent structural similarities either.

2.4 Cutnorm as Dissimilarity Measure

Given a matrix $A = (a_{ij})_{i \in R, j \in S}$, the cut norm is defined as

$$\|A\|_{\square} := \max_{I \subseteq R, J \subseteq S} \left| \sum_{i \in I, j \in J} a_{ij} \right| \quad (2.2)$$

The Cutnorm was first formulated by Freize and Kannan [4]. It was devised for their work on approximation algorithms for dense graph problems.

The Cut-Distance is then

$$d_{\square}(A, B) := \frac{1}{n^2} \|(A - B)\|_{\square} \quad (2.3)$$

Here we assume a reasonable measure preserving map is available and has transformed our matrices A, B into the same dimensions.

Alon and Noar has shown that the Cutnorm is MAXSNP hard via reduction from the MAXCUT problem [3] and also devised additional ρ -approximation algorithms for the Cutnorm.

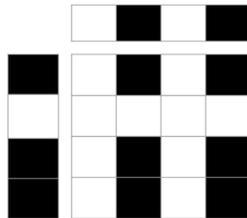


Figure 2.1: An illustration of Cutnorm vectors, the two vectors on the left and top of the matrix, and the Cutnorm elements of the matrix from the product of the two vectors

In Figure 2.1, we illustrate the Cutnorm pictorially. Consider the two vectors on the left and top of the matrix as the Cutnorm sets I, J . A black cell in the vector represents set membership within I, J . The black cells in the matrix represent all elements of $I \times J$.

These cells are the elements that will be summed over in the Cutnorm computation.

We also observe in (2.3) that a $1/n^2$ normalization is applied. This is established by previous literature from Lovasz and Freize. It imposes a restriction on the types of problems that are appropriate for the Cutnorm. Graphs that we measure to be dense. A dense graph is one where the number of edges connected to each node is a positive percentage of the total number of nodes. This ensures that as $n \rightarrow \infty$, the number of edges as a percentage does not converge to zero.

The thesis will present properties that the Cutnorm has on large and dense graph problems.

CHAPTER 3: CUTNORM APPROXIMATION

Alon and Noar [3] devised several randomized approximation algorithms that solves for I, J such that

$$\left| \sum_{i \in I, j \in J} a_{ij} \right| \geq \rho \|A\|_{\square} \tag{3.1}$$

where $\rho = 0.56$. This thesis has adopted one of Alon and Noar’s algorithms with a modification that allows for the use of a fast optimization algorithm. We introduce a new SDP solver by Wen and Yin [5]. This new SDP solver is used to find the solution to the $\|A\|_{\infty \rightarrow 1}$ SDP relaxation. We also introduce SVD to the rounding process for low rank adjacency matrices.

The rounding method that we will implement is one with lower bound but computationally feasible. It has $\rho = 0.27$.

3.1 Process of Approximation

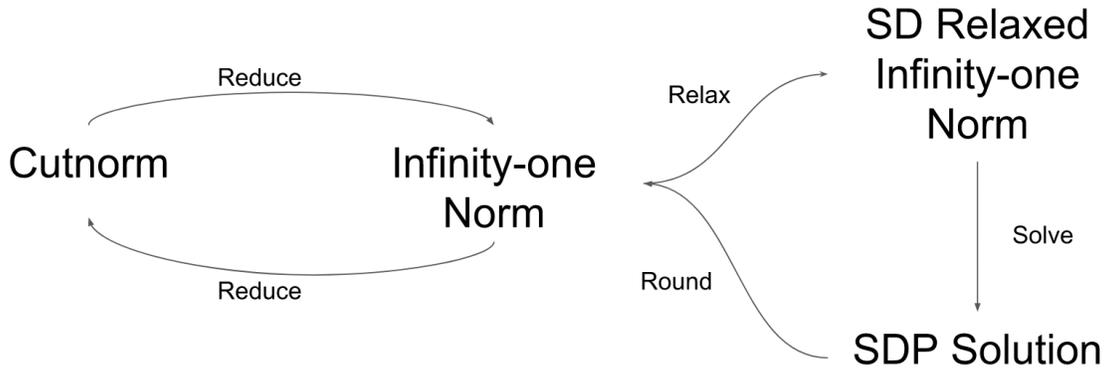


Figure 3.1: A diagram showing the overall process of approximating the Cutnorm

Figure 3.1 shows the sequence of reduction, relaxation, and rounding that is required

to compute the Cutnorm as detailed by Alon and Noar [3].

To understand the entire reduction process, we need to first understand why they are necessary.

3.2 Why the $\|A\|_{\infty \rightarrow 1}$?

Alon and Noar mentioned that it is “convenient” to study the $\|A\|_{\infty \rightarrow 1}$. We will discuss why it is so.

The formulation of $\|A\|_{\infty \rightarrow 1}$ is as follows

$$\|A\|_{\infty \rightarrow 1} = \max_{x_i, y_i \in \{-1, 1\}} \sum_{i=1}^n \sum_{j=1}^m a_{ij} x_i y_j \quad (3.2)$$

The $\|A\|_{\infty \rightarrow 1}$ optimizes over two binary vectors $\in \{-1, 1\}^n$. In the objective function, the element to be summed over is positive if $\text{sign}(x_i) = \text{sign}(y_i)$, and negative otherwise.

Consider, again, the Cutnorm problem formulation in (2.2)

$$\|A\|_{\square} := \max_{I \subseteq R, J \subseteq S} \left| \sum_{i \in I, j \in J} a_{ij} \right|$$

This Cutnorm optimization problem deals with set membership of vertices in the sets $I \subseteq R, J \subseteq S$. Another formulation of the Cutnorm problem that deals with the assignment of $\{0, 1\}$ to the vertex sets instead of set membership is the following

$$\|A\|_{\square} = \max_{x_i, y_i \in \{0, 1\}} \left| \sum_{i=1}^n \sum_{j=1}^m a_{ij} x_i y_j \right| \quad (3.3)$$

This new formulation of the Cutnorm is similar to the Infinity-One norm problem

with the exception of the binary assignment and the absolute value. The Infinity-One norm introduces an alternative problem to the Cutnorm such that the magnitude of the assignments are preserved, $|x_i|, |y_i| = 1$.

Further, it is shown by Alon and Noar that $\|A\|_{\square} \leq \|A\|_{\infty \rightarrow 1} \leq 4 \|A\|_{\square}$ and in the case of zero row sum and column sum, $\|A\|_{\infty \rightarrow 1} = 4 \|A\|_{\square}$.

3.3 Cutnorm Invariant Transformation

Alon and Noar has revealed that one can always augment the original matrix to create a zero row and column sum matrix that preserves Cutnorm. Therefore, one can always utilize the relationship $\|A\|_{\infty \rightarrow 1} = 4 \|A\|_{\square}$.

Since the proof was not explained in detail in the original paper, this thesis will dive into the details of how the Cutnorm is preserved.

Given a real value matrix $A = (a_{i,j})_{i \in R, j \in S}$, let A^* be an $(n+1) \times (m+1)$ matrix where for all $1 \leq i \leq n, 1 \leq j \leq m, a_{i,j}^* = a_{i,j}$, for all $1 \leq j \leq m, a_{n+1,j}^* = \sum_i -a_{i,j}$, for all $1 \leq i \leq n, a_{i,m+1}^* = \sum_j -a_{i,j}$, and $a_{n+1,m+1}^* = \sum_{i,j} a_{ij}$, $\|A\|_{\square} = \|A^*\|_{\square}$.

We show that this transformation preserves the Cutnorm.

Since A is a sub-matrix of A^* , $\|A\|_{\square} \leq \|A^*\|_{\square}$. Suppose, $\|A^*\|_{\square} = \left| \sum_{i \in I, j \in J} a_{ij} \right|$, where $I \subset [n+1], J \subset [m+1]$.

Let

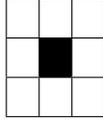
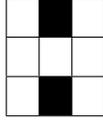
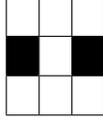
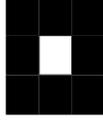
$$P = \begin{cases} [n+1] \setminus I, & \text{if } n+1 \in I \\ I, & \text{otherwise} \end{cases}$$

,

$$Q = \begin{cases} [m+1] \setminus J, & \text{if } m+1 \in J \\ J, & \text{otherwise} \end{cases}$$

.

Let us examine the possibilities of P, Q pictorially.

1. Given $I, J =$  then $P = I, Q = J$
2. Given $I, J =$  then $P = [n + 1] \setminus I, Q = J$
3. Given $I, J =$  then $P = I, Q = [m + 1] \setminus J$
4. Given $I, J =$  then $P = [n + 1] \setminus I, Q = [m + 1] \setminus J$

Since the sum of each row and column is zero, $\|A^*\|_{\square} = \left| \sum_{i \in I, j \in J} a_{ij} \right| = \left| \sum_{p \in P, q \in Q} a_{pq} \right|$. Furthermore, since $P \in R, Q \in S$, $\left| \sum_{p \in P, q \in Q} a_{pq} \right| \leq \|A\|_{\square}$. Therefore $\|A^*\|_{\square} \leq \|A\|_{\square}$, and we have $\|A^*\|_{\square} = \|A\|_{\square}$. ■

3.4 $\|A\|_{\infty \rightarrow 1}$ SDP relaxation

The $\|A\|_{\infty \rightarrow 1}$ can be relaxed to an SDP problem where an exact solution can be found to some error bound.

The $\|A\|_{\infty \rightarrow 1}$ SDP relaxation

$$\max_{u_i, v_i} \sum_{ij} a_{ij} u_i \cdot v_j \tag{3.4}$$

subject to $\|u_i\| = \|v_j\| = 1$

where $u_i, v_j \in \mathbb{R}^p$

The relaxation creates vectors $u_i, v_i \in \mathbb{R}^p$ instead of scalars $x_i, y_i \in \{-1, 1\}$ of the original $\|A\|_{\infty \rightarrow 1}$ problem. Further, there is an orthogonality constraint of $\|u_i\| = \|v_j\| = 1$ to ensure that the absolute value of $|x_i|$ is equal to the $\|u_i\| = 1$.

Since we have SDP solvers that can guarantee SDP solutions to an error bound of ϵ , we can solve for the solution to the SDP problem and round the solutions to our original $\|A\|_{\infty \rightarrow 1}$ problem to obtain a ρ -approximation algorithm.

3.5 Wen and Yin's Search Algorithm and Reduction to MAXCUT SDP relaxation

Wen and Yin's optimization algorithm [5] for problems with orthogonality constraints is well suited for our problem. Preservation of the constraints are expensive for orthogonality preserving optimization algorithms. Through the use of Cayley transformations and Crank-Nicolson update schemes, Wen and Yin formulated search algorithms that is much cheaper than existing methods.

The optimization algorithm solves the MAXCUT SDP relaxation exactly. This thesis uses a reduction from the $\|A\|_{\infty \rightarrow 1}$ SDP relaxation to the MAXCUT SDP relaxation.

The $\|A\|_{\infty \rightarrow 1}$ SDP relaxation in (3.4)

$$\begin{aligned} & \max_{u_i, v_i} \sum_{ij} a_{ij} u_i \cdot v_j \\ & \text{subject to } \|u_i\| = \|v_j\| = 1 \\ & \text{where } u_i, v_j \in \mathbb{R}^p \end{aligned}$$

We want to reduce the $\|A\|_{\infty \rightarrow 1}$ SDP problem to the MAXCUT SDP problem discussed in Wen & Yin's Paper

$$\begin{aligned} & \max_{V=[V_1, \dots, V_n]} \text{tr}(CV^T V) \\ & \text{subject to } \|V_i\| = 1, i = 1, \dots, n \end{aligned} \tag{3.5}$$

Using the trace identity $\sum_{ij} w_{ij} x_{ij} = \text{tr}(WX)$, this is equivalent to

$$\begin{aligned} & \max_{V=[V_1, \dots, V_n]} \sum_{ij} c_{ij} (V^T V)_{ij} & (3.6) \\ & \text{subject to } \|V_i\| = 1, i = 1, \dots, n \end{aligned}$$

By creating $V = [u_1, \dots, u_n, v_1, \dots, v_n]$ and $C = [0A; A0]$, for the case where $n = m$, we have the optimization problem

$$\begin{aligned} & \max_{V=[u_1, \dots, u_n, v_1, \dots, v_n]} \sum_{pq} a_{pq} u_p \cdot v_q + \sum_{pq} a_{pq} v_p \cdot u_q & (3.7) \\ & \text{subject to } \|u_i\| = \|v_i\| = 1, i = 1, \dots, n \end{aligned}$$

And for the case where A is symmetric and square, we have

$$\begin{aligned} & \max_{V=[u_1, \dots, u_n, v_1, \dots, v_n]} 2 \sum_{pq} a_{pq} u_p \cdot v_q & (3.8) \\ & \text{subject to } \|u_i\| = \|v_i\| = 1, i = 1, \dots, n \end{aligned}$$

Here we specified symmetric and square A . This is true for the adjacency matrices of simple graphs.

3.6 Choice of rank p for SDP relaxation

We have yet to define p , the vector length of $u_i, v_i \in \mathbb{R}^p$, in the $\|A\|_{\infty \rightarrow 1}$ SDP relaxation of (3.4). We can pick $p = n + m$ so that $\sum_{ij} a_{ij} u_i \cdot v_j$ is the min of the sdp [3]. Since the sdp solution is at least some constant away from $\|A\|_{\infty \rightarrow 1}$, we can guarantee a upper bound for the Cutnorm.

We choose $p = \max(\min(\text{round}(\sqrt{(2n)/2}), 100), 1)$ according to the suggested pa-

rameters in Wen & Yin's Paper for the max cut problem. Theorem 3 in Wen and Yin's paper states that a larger p is unnecessary for the MAXCUT sdg problem.

3.7 Gaussian Rounding

Let $u_i, v_j \in \mathbb{R}^p$ be solutions of the $\|A\|_{\infty \rightarrow 1}$ SDP relaxation. We would like to round the solution back to the solution of $\|A\|_{\infty \rightarrow 1}$.

Let g_1, g_2, \dots, g_p be standard independent Gaussian random variables and $G = (g_1, g_2, \dots, g_p)$.

$$x_i = \text{sign}(u_j \cdot G), y_j = \text{sign}(v_j \cdot G)$$

This rounding technique provides a solution that guarantees $\rho = \frac{4}{\pi} - 1 \approx 0.27$ approximation [3].

Since the solution of a ρ -approximation algorithm is bounded by $\rho OPT \leq f(x) \leq OPT$ for $\rho < 1$, we can repeat the rounding process and select the highest rounded solution. This becomes a statistics sampling problem. Given that we can repeat the rounding process to a sufficiently high number of iterations, we can obtain a sufficiently accurate result.

If our adjacency matrix, A , is low rank, we can consider computing an singular value decomposition first to reduce the rounding complexity.

Instead of the original rounding problem of

$$\|A\|_{\infty \rightarrow 1} \approx \sum_{i,j} C_{i,j} \text{sign}(g * u_i) \text{sign}(g * v_i) \tag{3.9}$$

Using low rank approximation of rank q for A into $a \in \mathbb{R}^{n \times q}, b \in \mathbb{R}^{q \times n}$

$$\|A\|_{\infty \rightarrow 1} \approx \sum_q \left(\sum_i a_{i,q} \text{sign}(g * u_i) \right) \left(\sum_j b_{j,q} \text{sign}(g * v_i) \right) \tag{3.10}$$

Rounding on the SVD rank q matrices results in $O(qn)$ per iteration as opposed to $O(n^2)$ via full rank matrices. The $O(n^3)$ cost of SVD has to be paid up front which brings the entire rounding operation with multiple iterations to $O(kqn) + O(n^3)$ instead of $O(kn^2)$ where k is the number of iterations. However, this is often justified as the rounding operations can be expensive over many rounding iterations. Naturally, this is only adequate if the adjacency matrix is sufficiently low rank.

After the SDP solutions are rounded to approximate solutions to the $\|A\|_{\infty \rightarrow 1}$ problem, we can obtain the $\|A\|_{\square}$ solution via the relationship $\|A\|_{\square} = \|A\|_{\infty \rightarrow 1} / 4$.

CHAPTER 4: CUTNORM PACKAGE

The Cutnorm approximation technique detailed in previous chapters has been implemented and hosted on the Python Package Index for public use. The *Cutnorm* package makes heavy use of Numpy operations for efficient vector and matrix operations. It also include several useful tools for calculating statistical significance and generating model graphs.

There is an existing Cutnorm package by David Koslicki [9] that utilizes the same technique from [3] but uses CSDP, a different SDP solver. Compared to Koslicki’s package, our Cutnorm package does not require separate binary installation and improves the running time via the use of Wen and Yin’s search algorithm. The computational times of Koslicki’s Cutnorm package and our Cutnorm package are shown in Figure 4.1. Same rounding iterations were used to measure the time of computation.

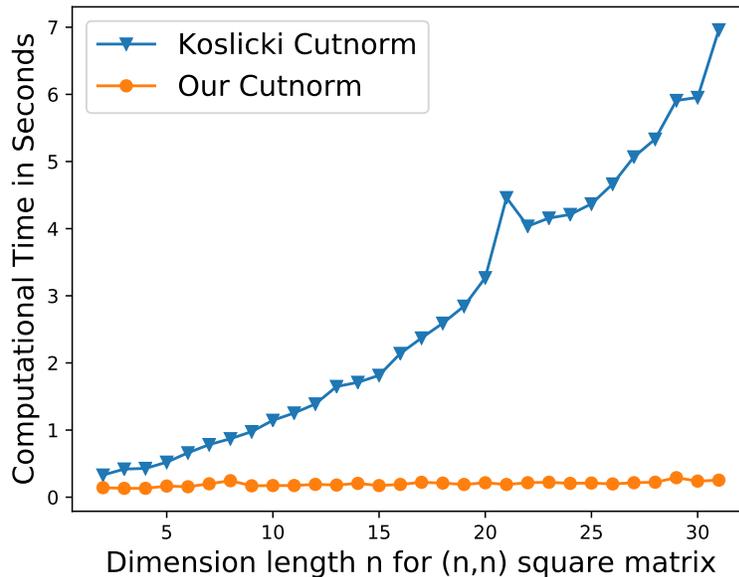


Figure 4.1: Computational Time for computing Cutnorm between two (n,n) square Erdős-Rényi Matrices

4.1 Complexity Analysis

Overall, the time complexity of the Cutnorm algorithm described in this thesis is $O(kn^2) + O(n^2)$ where k is the number of rounding iterations, or $O(kn \log n) + O(n^3)$ if SVD of rank $\log n$ is performed prior to the rounding process.

The Wen and Yin's proposed constraint preserving update scheme has computational complexity of $8np^2 + O(p^3)$. With our choice of p , this translate to $4n^2 + O(n^{3/2})$ which is of $O(n^2)$ per update. Since this update scheme uses a gradient method with line search on the matrix manifold, it has a linear convergence to the optimum [10]. Thus it converges in $O(\epsilon^{-1})$ iterations where ϵ is the tolerance. This translates to a total cost of $O(\frac{n^2}{\epsilon})$ or $O(n^2)$ if we consider constant tolerance.

Preprocessing of the input matrices such as the re-weighting and re-scaling of the input matrices is of $O(n^2)$. In each iteration of the rounding process, the rounding and computation of the cutnorm is of $O(n^2)$. Thus the overall Gaussian rounding process is $O(kn^2)$ for k iterations of rounding. If a $\log n$ low rank approximation of the difference matrix is performed prior to rounding, the rounding process will incur an additional $O(n^3)$ computational complexity for the SVD. However, this reduces the rounding time complexity for the iterations to $O(kn \log n)$.

In our experiments, the rounding operation can be the bottleneck of the algorithm. With the package default of 100 iterations, we find that the rounding time is several magnitudes higher than the sdp solver time. The user should decide how many rounding iterations is statistically sufficient and also determine if a low rank approximation of the underlying difference matrix is suitable. If the underlying difference matrix is low rank, then it is better to pay the up front cost of the SVD for faster rounding per iteration times. The debug flag can be set to true to generate additional information of the rounding process to determine the optimal parameters for the application.

The space complexity of the Cutnorm algorithm is of $O(n^2)$. If debug flag is set, some additional space of $O(kn)$ is used to store the computational information.

4.2 Installation

The Cutnorm package is hosted on PyPI, the Python Package Index. To install the package, run

```
1 pip install cutnorm
```

Currently, the package only supports Python3. All the relevant dependencies and files will be downloaded and installed.

4.3 Cutnorm Package Components

cutnorm.compute

The compute module contains the `compute_Cutnorm` function where Cutnorm between two matrices can be computed. `compute_cutnorm` supports weighted matrices and matrices of different dimensions. The `compute_Cutnorm` function solves the SDP problem with OptManiMulitBallGGB algorithm by Wen and Yin and performs Gaussian rounding on the SDP solution to obtain the Cutnorm.

cutnorm.OptManiMulitBallGGB

OptManiMulitBallGGB module contains the optimization algorithm by Wen and Yin. The original version released by Wen and Yin were in Matlab. We have translated the Matlab code to Python. Most of the original structure and variable names remain intact. In order to perform the reduction of Cutnorm to MAXCUT detailed in section 3.5, we have defined a new function, `Cutnorm_quad`, to compute objective function value and gradient for Cutnorm.

```

1 import numpy as np
2 from cutnorm import compute_cutnorm, tools
3
4 # Generate Erd\H{o}s-R\'enyi Random Graph
5 n = 100
6 p = 0.5
7 erdos_renyi_a = tools.sbm.erdos_renyi(n, p)
8 erdos_renyi_b = tools.sbm.erdos_renyi(n, p)
9
10 # Compute l1 norm
11 normalized_diff = (erdos_renyi_a - erdos_renyi_b) / n**2
12 l1 = np.linalg.norm(normalized_diff.flatten(), ord=1)
13
14 # Compute cutnorm
15 cutn_round, cutn_sdp, info = compute_cutnorm(erdos_renyi_a,
16 erdos_renyi_b)
17
18 print("l1 norm: ", l1) # prints l1 norm value near ~0.5
19 print("cutnorm rounded: ",
20 cutn_round) # prints cutnorm rounded solution near ~0
21 print("cutnorm sdp: ", cutn_sdp) # prints cutnorm sdp solution near ~0

```

Figure 4.2: Sample usage of package

cutnorm.tools

The tools module contains tools for making synthetic matrices like the Erdős-Rényi random graph and stochastic block models. This is a useful tool for understanding the advantages of Cutnorm and for testing the Cutnorm package.

4.4 Usage & Documentation

The package is simple to install and use. A sample usage of the package is shown in Figure 4.2. For additional usage and documentation of the source code, please visit <https://pingkoc.github.io/cutnorm/intro.html>.

CHAPTER 5: APPLICATIONS AND RESULTS

The thesis will explore several applications of using the Cutnorm to understand families of networks. Some of these applications are preliminary results. The objective is to show the ability of Cutnorm to provide accurate methods to analyzing families of networks.

The thesis has taken the approach of surveying different disciplines where dense networks exist as well as models of dense networks. In fact, one could even argue that dense networks can be extracted from problems of most disciplines given that one approaches the problem from a certain perspective. For an example, a social network of friendship is sparse. Our social interactions are very limited in the perspective of the global population. As the total population of the network tends to infinity, $n \rightarrow \infty$, each individual's handful of friends is almost non-existent in the global perspective. As we discussed earlier, a sparse network will result in vanishing Cutnorm as $n \rightarrow \infty$. Thus, applying Cutnorm techniques might not be adequate for social networks due to its sparsity. However, if we are curious about relationships among the attributes of each individual in the social network, we can extract dense networks from the relationship of these attributes. Beyond just friendship, given attributes such as age, gender, political affiliations, geolocation, and more, we can study relationships between people among these attributes. Our network then becomes naturally dense since everyone has these attributes, thus the number of connections between people does not diminish to zero as we increase the population.

5.1 Graph Models

Models of dense networks help us benchmark our algorithm against known solutions. Further, these models help to analyze the performance of the algorithms under different

model parameters.

Below we will explore the various norms of the difference matrix under various models. We compare the performance between various l_p norms and the Cutnorm.

5.1.1 Erdős-Rényi Model

The thesis introduced Erdős-Rényi Models (ER) in the earlier chapters. Here we revisit the Erdős-Rényi model to apply our Cutnorm package onto the models to measure performance. An Erdős-Rényi graph $G(n, p)$ is a graph with n vertices and each pair of vertices are connected with probability p .

Given two independent Erdős-Rényi random graphs with $p = 0.5$, we would like to compute the distance between the two graphs. The edit distance between the two graphs, the l_1 norm of the difference matrix, is 0.5 with large probability [7]. Since the difference matrix has elements that are $1/ - 1$ with probability of 0.25, and 0 with probability of 0.5. Thus the expected l_1 norm is 0.5 with high probability.

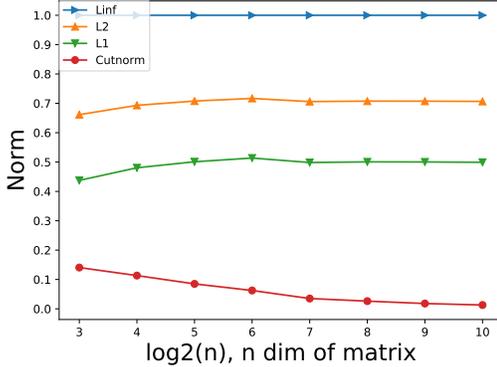
As discussed previously, the global structure of the two graphs are similar. In terms of graph convergence as $n \rightarrow \infty$, the two graphs converges to the same graphon. Thus the l_1 distance fails to give an accurate notion of global structural similarity.

The Cutnorm should be able to provide a good solution to the distance. We observe in Figure 5.1a that as n increases, the Cutnorm approaches 0 while the l_1 norm approaches 0.5. The other l_p norms approach other constants that are non-zero.

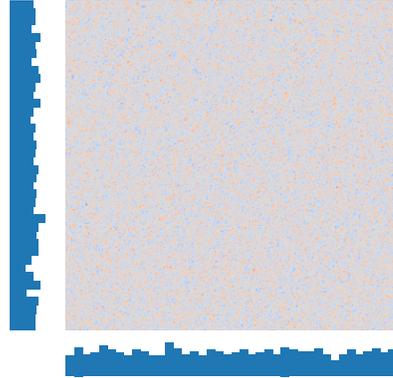
We also show that $\square \leq l_1 \leq l_2 \leq l_\infty$ which matches previously established theory.

Further, we can examine the Cutnorm sets. The Cutnorm set is the underlying sets I, J that generate the Cutnorm as shown in (2.2).

In Figure 5.1b, we show the Cutnorm sets on the difference matrix between two ER graphs. For purposes of illustration, I have binned the elements in the set. At a glance, we see that the Cutnorm sets are uniform. Uniformity in the Cutnorm sets



(a) l_p norms and Cutnorm on the difference matrix



(b) Cutnorm set on the difference matrix

Figure 5.1: Cutnorm and Cutnorm sets on the difference matrix of two independent Erdős-Rényi $p = 0.5$ random graphs.

mean that no edge sets I, J strongly define the maximum of the Cutnorm. In other words, the Cutnorm sets isolate structural differences between the two independently generated ER graphs. Since the expectation of the summation of the elements of the difference matrix is 0, regardless of which Cutnorm sets are generated, the expectation of the summation has expectation of 0.

5.1.2 Erdős-Rényi $p = 0.5$ and Stochastic Block Model

Here we introduce another model, the Stochastic Block Model (SBM). SBMs have been studied and used as a generalization to Erdős-Rényi models [11], [12]. Stochastic Block Matrices are matrices with block regions that have a specific edge generating probability p . Given row set \mathbf{R} and column set \mathbf{C} , partition the sets into $\mathbf{R} = R_1 \cup \dots \cup R_I$, $\mathbf{C} = C_1 \cup \dots \cup C_J$, and generate elements with probability $P_{i,j}$ for every region in $\mathbf{R} \times \mathbf{C}$.

Figure 5.2 shows an example with row and column partitioned into three sets. Each region defined by the row and column have a probability $P_{i,j}$ of generation.

| | | |
|-----------|-----------|-----------|
| $P_{1,1}$ | $P_{1,2}$ | $P_{1,3}$ |
| $P_{2,1}$ | $P_{2,2}$ | $P_{2,3}$ |
| $P_{3,1}$ | $P_{3,2}$ | $P_{3,3}$ |

Figure 5.2: An example of SBM with row and column partitioned into three sets

| | |
|-----|-----|
| 0.5 | 0.5 |
| 0.5 | 0.5 |

| | |
|-----|-----|
| 0.5 | 0.5 |
| 0.5 | 1.0 |

- (a) The edge generating probability matrix of the ER model. All of the regions have the same edge generating probability.
- (b) The edge generating probability matrix of SBM. Each element of the matrix represents the edge generating probability.

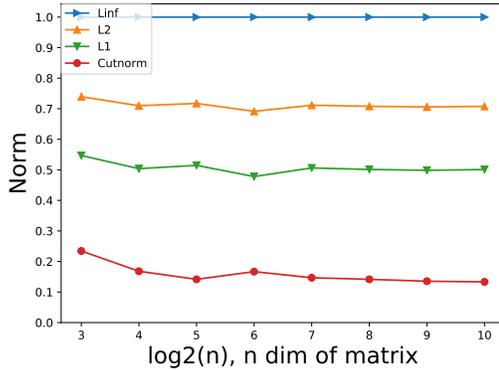
Figure 5.3: The generating models for ER $p = 0.5$ and a specially defined SBM. The difference matrix and the norms on the difference matrix will be computed against the two graphs.

Recall Lovasz's definition of a step function $W \in \mathcal{W}$ as a subset of the graphon limit object \mathcal{W} . The SBM is a finite definition for W . Consider the general model of Erdős-Rényi random graphs and the SBM as $n \rightarrow \infty$, while the Erdős-Rényi graphs converges to a graphon limit object of p everywhere, the SBMs converges to a step function defined by $p_{i,j}$ of the region.

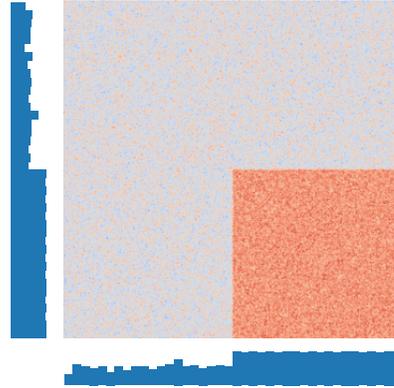
SBMs are valuable to our study because it gives us a more general stochastic graph model beyond the simple Erdős-Rényi graphs while still being very simple to define.

Given one Erdős-Rényi graph with $p = 0.5$ and a Stochastic Block Model with $p = 0.5$ for three quadrants and $p = 1.0$ for one quadrant, we would like to find the distance between the two graphs. Figure 5.3 illustrates the generating probability matrix for the two models. We can see that the generating probability is identical in most regions except one of the quadrants. The difference matrix $D = d_{ij}$ has three quadrants with $Pr(d_{ij} = 1) = 0.25$, $Pr(d_{ij} = -1) = 0.25$, and $Pr(d_{ij} = 0) = 0.50$, one quadrant with $Pr(d_{ij} = 1) = 0.50$, and $Pr(d_{ij} = 0) = 0.50$.

Compare these two graphs to the two independent Erdős-Rényi random graphs case, we can see that there is some global structural difference between the two graphs now.



(a) l_p norms and Cutnorm on the difference matrix



(b) Cutnorm set on the difference matrix

Figure 5.4: Cutnorm and Cutnorm sets on the difference matrix of an Erdős-Rényi $p = 0.5$ random graph and a Stochastic Block Model graph where edges are independent and uniformly random with $p = 0.5$ for 3 out of 4 quadrants and $p = 1$ for 1 out of 4 quadrants.

As n increases, with high probability, the l_1 norm should still be 0.5. This indicates that the l_1 norm doesn't tell us anything interesting beyond that the l_1 distance between the two is the same as the l_1 distance in the case where the two are both ER. The Cutnorm on the other hand should converge to 0.125 as n gets large. It should be 0.125 since it is the sum of the difference in the quadrant that has different edge generating probability between the two models. The Cutnorm gives an accurate representation of the global structural difference for the examples studied here.

In Figure 5.4a, we confirm our hypothesis that the l_1 norm still converges to 0.5 but the Cutnorm converges to 0.125. The other norms converges identically to the case when two graphs were ER with same probability.

Figure 5.4b shows the Cutnorm sets on the difference matrix. Unlike the case of the two ER graphs, the Cutnorm sets here show a distribution that highlights the region where the two graphs are structurally different. Thus by examining the Cutnorm sets,

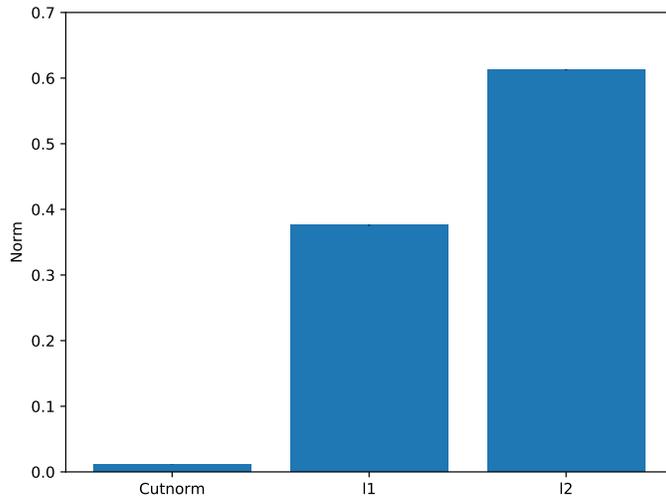


Figure 5.5: Cutnorm between two isomorphic graphs but with one graph’s adjacency matrix representation shifted by one position. Error bars are not visible.

we can identify the structural differences between the two graphs.

5.2 Misregistration/Shifting of Vertex Labeling

It is reasonable to assume that vertices get mislabeled or the labels are shifted during a data extraction process. A robust norm should not penalize minor mislabeling or the misalignment of the nodes between two adjacency matrices.

Given two identical Erdős-Rényi graphs with $p = 0.5$. If we consider the Cutnorm or the l_p norm of the two graphs, the norm values should be 0 since the two are identical. However, we will take one of them and shift it by rolling the adjacency matrix diagonally by 1 position. The underlying graphs are still isomorphic to each other but adjacency matrix representation has simply changed.

Figure 5.5 shows the Cutnorm of the difference matrix. The l_p norms increase drastically while the Cutnorm stays relatively robust to this perturbation.

While this is an example with just ER graphs, we believe that the robustness generalizes to other types of graphs.

5.3 NEUROIMAGING

The human connectome is the map of neural connections in the human brain. Connectomes are believed to be responsible for our mental and physical functions and are the foundations of neuroscience [13]. However, the human connectome is large and the full mapping requires substantial scientific advancements and computational power that is not yet available. To give a perspective on the progress of Connectome research, only few animals have had their Connectomes completely mapped out. Roundworm *C. Elagans* was the first animal to have its Connectome mapped out in 1986 [14] and it has 302 neurons. The human brain is estimated to have approximately 86 billion neurons [15].

There are several projects aimed at accelerating the effort to map the human brain. Among those, several take an open-data approach to release functional Magnetic Resonance Imaging (fMRI) timeseries data of volunteers. They include, but are not limited to, *1000 Connectome Project* [16], *MyConnectome Project* [17], and *Human Connectomes Project* [18]. These projects hosts initiatives for global collection of fMRI scans. These fMRI scans are typically timeseries that can be passed through many data processing pipelines that extract signals from the images. The signals extracted can be mapped to existing atlases or brain parcellation that segment the brain into regions.

The fMRI analysis pipelines typically separated into various stages (illustrated in Fig. 1 of [19]): ROIs estimation, time series extraction, matrix estimation, and classification. The ROIs estimation stage is where regions of interests are extracted from the brain. Sometimes the regions of interests are referred to as an atlas. Different parcellation of the brain highlight different signals. Time series extraction is where the fMRI images are analyzed for activation within the ROIs. For an example, given an atlas of 200 ROIs, signals from each of the ROIs are extracted through time. We

will have a $200 \times t$ timeseries data where t is the number of fMRI images in a fixed amount of time. Matrix estimation generates a matrix that represents the connectivity between different ROIs. This connectivity can be estimated through several methods such as Pearson Correlation, Fisher correlation, partial correlation, tangent embedding, and covariance. Abraham et al. [19] has shown that tangent embedding has good results for the ABIDE autism dataset. The classification stage can utilize any machine learning model such as Support Vector Classification, Artificial Neural Networks, and more.

This thesis focuses on the *ADHD200* [20] dataset released by the *1000 Connectome Project*. The *ADHD200* project managed to collect 973 fMRI brain scans from Attention Deficit Hyperactivity Disorder (ADHD) patients as well as control group. It was a collaboration among several universities and research institutes around the world.

A competition was hosted in 2011 with the goal to learn a machine learning classifier that can predict individuals into three categories: ADHD-Combined, ADHD-Inattentive, and healthy control. The competition was actually won by a team that used only the phenotypic data of the participants without any of the fMRI scans [21]. The team achieved a diagnostic accuracy of 62.52%. At the time, the other teams that used the fMRI scans for prediction were not able to achieve such accuracy. Better methods of prediction and analysis of the data are still being sought after.

This thesis will attempt to use the Cutnorm to analyze distances between various brain structures using results from existing data processing pipelines. The representation of brain connectivity as correlation matrices or tangent embedding is dense and suitable for the use of the Cutnorm.

The Athena processing pipeline was used to prepare the brain parcellation for the thesis. In particular, we used the *CC200* brain parcellation timeseries for its finer-grained ROIs. We followed the additional processing procedures in [19] to generate

tangent embedding of the fMRI data. Each tangent embedding is a connectivity graph of a brain.

We would like to investigate whether ADHD brains are structurally different to healthy brains in terms of the Cut-Distance. Unlike the competition, we will simply deal with the binary classification problem of ADHD/Healthy brain classification. To do so, we take the tangent embedding graph and compute pairwise Cut-Distance among the participants. This gives us a pairwise distance matrix between all of the participants. We then take the distance matrix and compute a TSNE embedding to reduce the dimensionality to 2 and plot the results. Distance Based F-Test [22] is also computed on the computed Cutnorm pairwise distance matrix to get a quantitative view of the separability of the individuals.

However, we need to extract the structures that are significant. The timeseries and the tangent embedding graph contains structures that are not relevant to the classification of ADHD and healthy brains. Since we view the Cut-Distance as a metric for defining distances among graph structures, we need to suppress the structures that are not important and amplify the structures that are. The optimum weights to the difference matrix can be defined as the weights that maximizes the Cutnorm for identically labeled individuals. For a binary classification task, the optimum weight matrix W

$$\operatorname{argmin}_W \sum_{a \in P} \sum_{b \in P} (y_a y_b) \|W(S_a - S_b)\|_{\square}$$

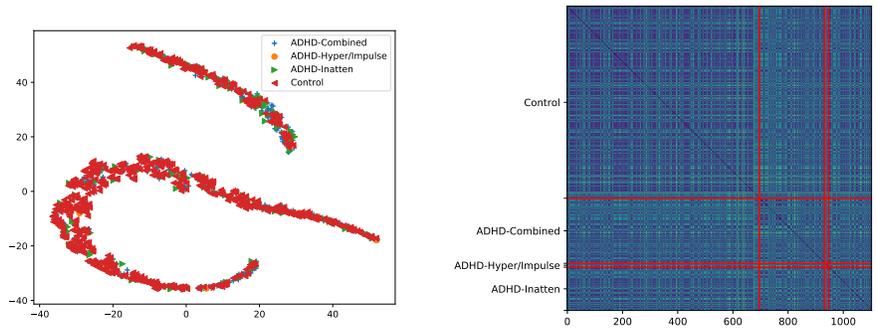
where $y_a, y_b \in \{1, -1\}$ are ADHD/Healthy labels and S_a, S_b are the tangent embedding graphs of two individuals among the population set P . For individuals with the same labels, $y_a y_b$ is positive and the Cutnorm should be minimized. For those with differing labels, $y_a y_b$ is negative and the Cutnorm should be maximized.

Since the Cutnorm is not easily differentiable, we can use other norms or methods to approximate this optimization problem. One such example is the Frobenius norm which can be solved exactly or approximated. Another method is to train a l_2 regularized SVM on the difference matrix of the tangent embedding. Since the SVM finds margin maximizing separating hyperplane on the dataset, the normal should be suitable as weights of the difference matrix of tangent embedding. Multiplying the SVM coefficients to the elements of the difference matrix is equivalent to computing the distance between two features with respect to the normal of the separating hyperplane. We used the SVMs to approximate the optimum weights.

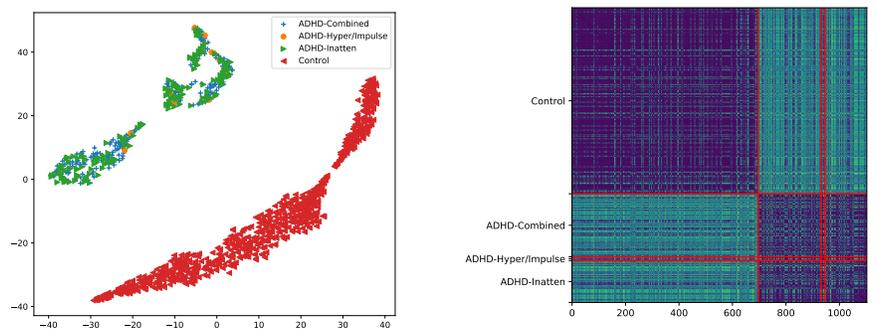
The SVMs trained with the entire dataset on the lower triangular elements of the tangent embedding has training accuracy of 1.0. Thus the dataset is fairly linearly separable and the SVM is able to learn a separating hyperplane that can separate the training data perfectly. We then performed Stratified 3-Fold cross validation on the SVM and obtained cross validation scores [0.6653, 0.6914, 0.5960] and an average of 0.6509. This indicates that training on 2/3 of the data does not generalize well to testing on the 1/3 of the data. A larger dataset might alleviate this problem but obtaining these 973 fMRI scans already took a lot of collaborative effort.

We have experimented with both weighted tangent embedding and non-weighted tangent embedding. Figure 5.6 shows the TSNE embedding of the pairwise Cut-Distance matrix and the matrix itself. We observe that the sorted pairwise Cut-Distance in Figure 5.6b does not display clear block structure. Block structures indicate similarity in Cut-Distance for subsets of the vertices (individuals in this case). Perhaps also due to the lack of block structure, the TSNE embedding of the matrix fails to display clear clustering between the two classes.

However, after we apply the SVM weights on the graphs, we start to see clear structure in pairwise Cut-Distance matrix in Figure 5.6d. Further, the TSNE embedding



(a) TSNE embedding of pairwise Cut-Distance without SVM weights between the different individuals. (b) Pairwise Cut-Distance matrix without SVM weights sorted by the labels on the individuals



(c) TSNE embedding of pairwise Cut-Distance with SVM weights between the different individuals. (d) Pairwise Cut-Distance matrix with SVM weights sorted by the labels on the individuals

Figure 5.6: Pairwise Cut-Distance matrix and the TSNE embedding of tangent embedding with and without weights learned from SVM.

of the Cut-Distance matrix also shows clear clustering among the two classes. For illustrative purposes, I have also indicated the multiple types of ADHD, but structural differences among the various ADHD measures are not clearly visible.

The thesis also computed DBF testing on the pairwise distance matrices of different data processing pipelines. DBF testing offers a p-value to the difference in the distribution of the two groups (in our case, ADHD and control) based on pairwise distances. A low p-value (≤ 0.05 or ≤ 0.01) indicates strong inconsistency with the null hypothesis that supposes the two classes are indistinguishable. Thus given a low p-value, we

| Matrix Estimation Type | p-value |
|---|-----------------------|
| <i>Pearson Correlation</i> | 9.26×10^{-1} |
| <i>Covariance</i> | 8.94×10^{-3} |
| <i>Tangent Embedding</i> | 8.71×10^{-8} |
| <i>Tangent Embedding with SVM weights trained on 66% of data</i> | 0.00 |
| <i>Tangent Embedding with SVM weights trained on 100% of data</i> | 0.00 |

Figure 5.7: DBF Statistic Testing p-values for separation using pairwise distance matrix under different matrix estimation methods.

can affirm that the two classes are clearly distinguishable. The thesis has computed pairwise distances using various matrix estimation techniques: Pearson Correlation, Covariance, Tangent Embedding, Tangent Embedding with SVM weights trained on 100% of the data, and Tangent Embedding with SVM weights trained on 66% of the data. Of these methods, we have only shown the TSNE embedding of the Tangent Embedding with SVM weights in Figures 5.6. Figure 5.7 shows DBF p-values on the pairwise distance matrix under various matrix estimation methods. We can observe that besides Pearson Correlation, the other matrix estimation techniques display low p-values. Given tangent embedding of the ROI timeseries data of ADHD and Control individuals, one can compute the pairwise Cut-Distance that can be separated into two groups up to some significance level. In particular, we see that Tangent embedding with the SVM weights show the lowest p-value, indicating that the two groups can be perfectly separated given this method of matrix estimation.

5.4 ARTIFICIAL NEURAL NETWORK REPRESENTATION LEARNING

Consider training Artificial Neural Networks (NN) beyond just minimizing loss functions, improving accuracy, or optimizing some other metric, but learning representation of the dataset. This view of the NN is very appropriate for Autoencoders

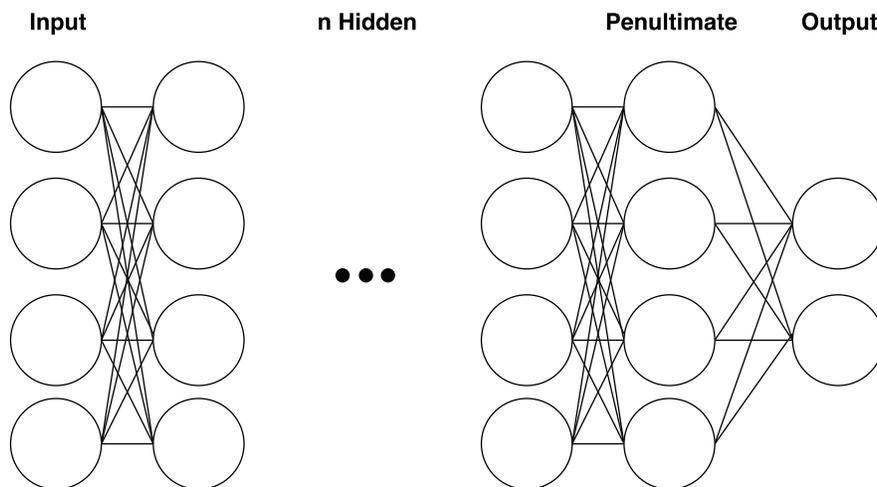


Figure 5.8: An example of a simple feed-forward Neural Network with fully connected layers, n hidden layers, and 2 output nodes.

and Convolutional Neural Networks on images. Autoencoders learn a compressed encoding of the data while attempting to minimize the loss of data during the decoding phase. Convolutional Neural Networks on images extract image features.

Li et al. has studied convergent learning in deep neural networks [23]. They looked at whether features learned by different architectures are largely the same. To do so, they studied bipartite matching between the learned representations of different architectures.

We are similarly interested in studying the learned representation of a deep neural network. However, instead of studying bipartite matching, we look at how the Cutnorm can be used to give a distance between the learned representations.

See Figure 5.8 for an example of a simple feed-forward NN. As data passes through the NN, the data representation changes. With non-linear activation functions on each of the NN nodes, the data goes through non-linear transformations as it passes through each layer. Consider the penultimate and the ultimate layer. The output of the ultimate layer depends on the problem of interest: binary/multi-class classification, estimation, or multi-label classification. The penultimate layer outputs a representation

of the data such that the ultimate layer can easily prepare into a format that is suitable for the problem. We can think of the penultimate layer output as a final learned representation of the data.

For an example, the NN architecture in Figure 5.8 might be used for a binary classification task. The penultimate layer might learn a representation to ensure that the output layer can use a simple linear regression to separate the two classes.

We would like to investigate the convergence of the learned representation. To train a NN network, the updates are propagated from the output layer towards the input layers. Each update is an attempt to minimize some loss function or to change the representation of the data so that the loss function is minimized. Therefore given some convex loss function, we should observe the convergence of the loss through the updates. The convergence of the loss should also be reflected on the data representation. If the loss is being minimized each epoch, the updates should also be minimized. Minimizing the updates means the data representation should converge.

To estimate data representation, we compute Pearson correlation between features of the data. This is an idea borrowed from Neuroimaging where correlation between different ROI represents functional connectivity.

Given n data samples $x_i \in \mathbb{R}^d, 1 \leq i \leq n$, we will capture the representation at the penultimate level as $z_i \in \mathbb{R}^k$ (k depends on the NN structure) and construct a representation of the topology as $A = a_{ij}$ where $a_{ij} = \text{corr}(z_i, z_j)$, the Pearson correlation between the two data points.

Let A^t be the data representation at the t 'th epoch of the NN training process. By considering $\|A^0 - A^t\|_{\square}$, where A^0 is the original dataset, we would like to determine if the Cut-Distance between the original dataset and the learned representation at time t converges to some value as $t \rightarrow \infty$.

We would also like to investigate how the number of hidden layers affect the data

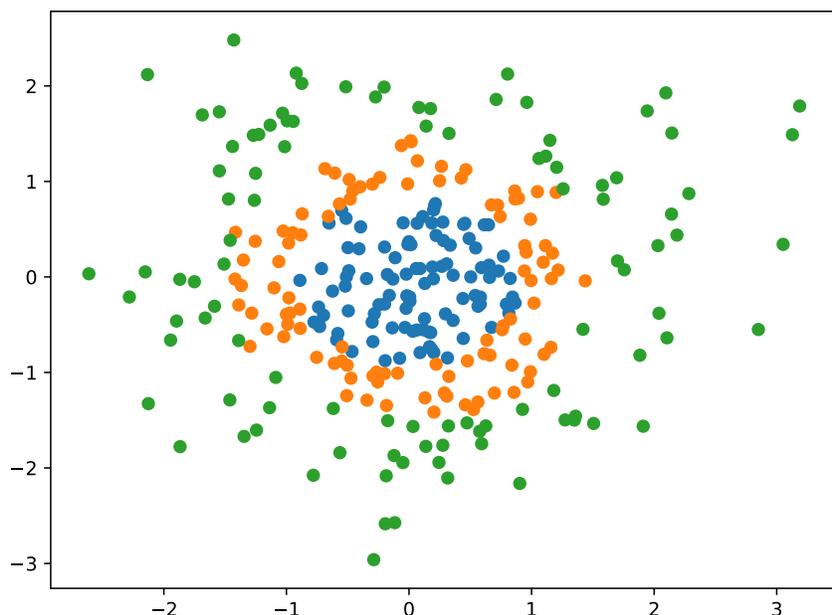
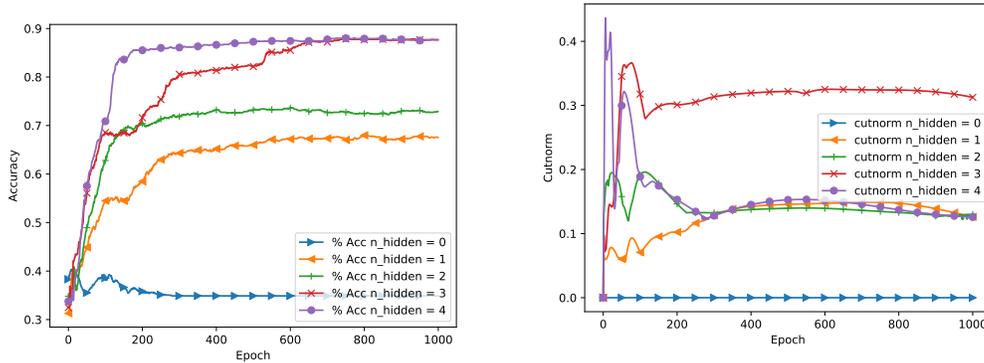


Figure 5.9: An example of Gaussian quantiles of 2-Dimension with 3 classes.

representation convergence rate. To achieve this, we need to find a dataset that is non-linear. Consider each hidden layer of NN with a non-linear activation as adding additional non-linear transformation to the data representation. Given some non-linear data, a NN with no hidden layers should not achieve high accuracy since only linear transformations were performed. A NN with more number of hidden layers should be able to successfully transform the data representation so that the data is more linearly separable as it gets to the penultimate layer. We will test this hypothesis and also visualize the data representation through the use of Cutnorm.

To perform this experiment, we need to obtain some non-linear data. One such dataset is the Gaussian quantiles. We generate n -dimensional Gaussian quantiles with k -classes. Each class occupies a certain n -dimensional spherical shell. See Figure 5.9 for an example.

We wish to train a NN to classify a 10-dimensional 3-classes Gaussian quantile dataset. PyTorch was used as the NN framework. The NN architecture consists of an



(a) Accuracy of NN of various number of hidden layers across epoch of the training data. The vertical axis is accuracy and the horizontal axis is the number of epochs. (b) Cutnorm between the original data representation and the i th epoch data representation. The vertical axis is the Cut-Distance and the horizontal axis is the number of epochs.

Figure 5.10: Accuracy and Cutnorm results from NN trained on 10-dimensional 3-class Gaussian quantile dataset.

input layer that has 10 nodes, n hidden layers with 10 nodes with tanh activation at each node, and two output nodes. BCEWithLogitsLoss was used as our loss function and Adam was used as the optimizer. Therefore with the number of hidden layers as 0, the NN acts as Logistic Regression.

Figure 5.10a shows the accuracy through training epochs. We can observe that epochs to convergence is reduced as the number of hidden layers is increased. This does not factor in the higher computational costs of a deeper NN. Deeper NN is able to find non-linear transformations given much fewer data epochs.

Figure 5.10b shows the Cutnorm between the original data representation, A^0 , and the data representation at the t 'th epoch, A^t . We observe that the Cutnorm converges to some value at around the 200'th epoch. Notice this is also where the accuracy starts to converge in Figure 5.10a. The initial 0 – 200 epochs was a phase with high fluctuations in Cutnorm. This is probably the period when the largest updates are applied. Adam is an adaptive optimizer, thus it is expected that large

updates are performed during the initial stages of the optimization [24]. The final value of convergence for Cutnorm in Figure 5.10b is not extremely informative. Since there may be multiple data representations that has similar Cut-Distance away from the original data representation.

To see if the same representation is learned, we need to compare the learned representation against the different NN architectures. This relates to the Measure Preserving Maps discussed in previous chapters. We need Measure Preserving Maps between the data representation learned by the different NN architectures. A brute force approach would be to attempt all $O(10!)$ permutations. Li et al. [23] has devised a method to compute Bipartite matching between the learned representation of CNN on images with great results. Since the thesis does not deal with Measure Preserving Map in detail, this was not attempted.

The results from the analysis of data representation convergence shows that there exists an additional method of analyzing convergence of NN training that is beyond loss functions and the other existing metrics. The approach of directly monitoring the convergence of data representation sheds light onto how different NN architectures transform the dataset.

CHAPTER 6: DISCUSSION AND CONCLUSION

This thesis presents the process to approximate the Cutnorm and its applications. We have taken the algorithm detailed by Alon and Noar and applied Wen and Yin's search algorithm on it. We believe that this approach is computationally faster compared to existing solutions.

Large graphs exist in many problem domains, the thesis explored synthetic models, Neuroimaging problems, and Artificial Neural Network data representation learning.

Through the survey of graph convergence and limits, we gain an additional perspective on graphs that scale and the distance between them. This is especially applicable in the context of the human connectome. As discussed, the human connectome is estimated to have 86 billion neurons. An fMRI scan provides a 3D representation of the brain under a fixed resolution. Finer-grained brain parcellation can be used to map out smaller regions of the brain, but, with the current technology, that is still not at the scale of 86 billion neurons. Every brain parcellation is taking a sample of the brain at a lower resolution. We would expect that as our technology advances, we will have brain parcellation that are much higher resolution and we will have the computational power to power such parcellation data. Therefore our discussion on the convergence of functional connectivity of brain parcels to a graphon limit object becomes appropriate. Given the brain parcellation data we have currently, we can compare the graphs under the knowledge that these graphs converge to some graphon limit object.

The thesis has also taken the Cutnorm to theoretical models such as the Erdős Rényi random models and the Stochastic Block models to verify our assumptions regarding the Cut-distance between these graphs. We observed that for graphs that converge to some graphon limit object, the Cutnorm approaches the expected value as $n \rightarrow \infty$.

Although this thesis has only addressed two practical disciplines for the use of Cut-

norm, we believe that there are many more interesting problems where it is applicable. The Cutnorm is useful in analyzing graph distances given that the graphs in questions belong to a family of graphs and sufficiently dense.

The Cutnorm package detailed in this thesis is open source and available for quick installation and public use.

REFERENCES

- [1] L. Lovász, *Large Networks and Graph Limits*, ser. American Mathematical Society colloquium publications. American Mathematical Society, 2012, ISBN: 978-0-8218-9085-1. [Online]. Available: <https://books.google.com/books?id=FsFqHLid8sAC>.
- [2] C. Borgs, J. T. Chayes, L. Lovász, V. T. Sós, and K. Vesztegombi, “Convergent sequences of dense graphs I: Subgraph frequencies, metric properties and testing,” *Advances in Mathematics*, vol. 219, no. 6, pp. 1801–1851, Dec. 2008, ISSN: 0001-8708. DOI: 10.1016/j.aim.2008.07.008. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0001870808002053> (visited on 02/12/2018).
- [3] N. Alon and A. Naor, “Approximating the Cut-Norm via Grothendieck’s Inequality,” en, *SIAM Journal on Computing*, Jul. 2006. DOI: 10.1137/S0097539704441629. [Online]. Available: <http://epubs.siam.org/doi/10.1137/S0097539704441629> (visited on 02/17/2018).
- [4] A. Frieze and R. Kannan, *Quick Approximation to Matrices and Applications*. 1998.
- [5] Z. Wen and W. Yin, “A feasible method for optimization with orthogonality constraints,” en, *Mathematical Programming*, vol. 142, no. 1-2, pp. 397–434, Dec. 2013, ISSN: 0025-5610, 1436-4646. DOI: 10.1007/s10107-012-0584-1. [Online]. Available: <https://link.springer.com/article/10.1007/s10107-012-0584-1> (visited on 02/17/2018).
- [6] L. Lovász and B. Szegedy, “Limits of dense graph sequences,” *Journal of Combinatorial Theory, Series B*, vol. 96, no. 6, pp. 933–957, Nov. 2006, ISSN: 0095-

8956. DOI: 10.1016/j.jctb.2006.05.002. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0095895606000517> (visited on 03/29/2018).
- [7] L. Lovasz, “Very large graphs,” *ArXiv:0902.0132 [math]*, Feb. 2009, arXiv: 0902.0132. [Online]. Available: <http://arxiv.org/abs/0902.0132> (visited on 01/24/2018).
- [8] P. Erdos and A. Renyi, “On the evolution of random graphs,” *Publ. Math. Inst. Hung. Acad. Sci.*, vol. 5, pp. 17–61, 1960.
- [9] D. Koslicki, *CutNorm: Approximation of the cut norm distance via a semidefinite relaxation*, original-date: 2016-02-04T07:23:25Z, Feb. 2016. [Online]. Available: <https://github.com/dkoslicki/CutNorm> (visited on 04/03/2018).
- [10] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Jan. 2008, vol. 78, ISBN: 978-0-691-13298-3. DOI: 10.1515/9781400830244.
- [11] E. Mossel, J. Neeman, and A. Sly, “Stochastic Block Models and Reconstruction,” *ArXiv:1202.1499 [math-ph]*, Feb. 2012, arXiv: 1202.1499. [Online]. Available: <http://arxiv.org/abs/1202.1499> (visited on 04/06/2018).
- [12] E. Abbe, “Community detection and stochastic block models: Recent developments,” *ArXiv:1703.10146 [cs, math, stat]*, Mar. 2017, arXiv: 1703.10146. [Online]. Available: <http://arxiv.org/abs/1703.10146> (visited on 04/06/2018).
- [13] O. Sporns, G. Tononi, and R. Kötter, “The Human Connectome: A Structural Description of the Human Brain,” en, *PLOS Computational Biology*, vol. 1, no. 4, e42, Sep. 2005, ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.0010042. [Online]. Available: <http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.0010042> (visited on 04/05/2018).

- [14] J. G. White, E. Southgate, J. N. Thomson, and S. Brenner, “The structure of the nervous system of the nematode *Caenorhabditis elegans*,” en, *Phil. Trans. R. Soc. Lond. B*, vol. 314, no. 1165, pp. 1–340, Nov. 1986, ISSN: 0080-4622, 2054-0280. DOI: 10.1098/rstb.1986.0056. [Online]. Available: <http://rstb.royalsocietypublishing.org/content/314/1165/1> (visited on 04/05/2018).
- [15] F. A. C. Azevedo, L. R. B. Carvalho, L. T. Grinberg, J. M. Farfel, R. E. L. Ferretti, R. E. P. Leite, W. Jacob Filho, R. Lent, and S. Herculano-Houzel, “Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain,” eng, *The Journal of Comparative Neurology*, vol. 513, no. 5, pp. 532–541, Apr. 2009, ISSN: 1096-9861. DOI: 10.1002/cne.21974.
- [16] M. Mennes, B. B. Biswal, F. X. Castellanos, and M. P. Milham, “Making data sharing work: The FCP/INDI experience,” *NeuroImage*, vol. 82, pp. 683–691, Nov. 2013, ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2012.10.064. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1053811912010671> (visited on 04/05/2018).
- [17] R. A. Poldrack, T. O. Laumann, O. Koyejo, B. Gregory, A. Hover, M.-Y. Chen, K. J. Gorgolewski, J. Luci, S. J. Joo, R. L. Boyd, S. Hunicke-Smith, Z. B. Simpson, T. Caven, V. Sochat, J. M. Shine, E. Gordon, A. Z. Snyder, B. Adeyemo, S. E. Petersen, D. C. Glahn, D. R. McKay, J. E. Curran, H. H. H. Göring, M. A. Carless, J. Blangero, R. Dougherty, A. Leemans, D. A. Handwerker, L. Frick, E. M. Marcotte, and J. A. Mumford, “Long-term neural and physiological phenotyping of a single human,” en, *Nature Communications*, vol. 6, p. 8885, Dec. 2015, ISSN: 2041-1723. DOI: 10.1038/ncomms9885. [Online]. Available: <https://www.nature.com/articles/ncomms9885> (visited on 04/05/2018).

- [18] D. C. Van Essen, K. Ugurbil, E. Auerbach, D. Barch, T. E. J. Behrens, R. Bucholz, A. Chang, L. Chen, M. Corbetta, S. W. Curtiss, S. Della Penna, D. Feinberg, M. F. Glasser, N. Harel, A. C. Heath, L. Larson-Prior, D. Marcus, G. Michalareas, S. Moeller, R. Oostenveld, S. E. Petersen, F. Prior, B. L. Schlaggar, S. M. Smith, A. Z. Snyder, J. Xu, E. Yacoub, and WU-Minn HCP Consortium, “The Human Connectome Project: A data acquisition perspective,” eng, *NeuroImage*, vol. 62, no. 4, pp. 2222–2231, Oct. 2012, ISSN: 1095-9572. DOI: 10.1016/j.neuroimage.2012.02.018.
- [19] A. Abraham, M. P. Milham, A. Di Martino, R. C. Craddock, D. Samaras, B. Thirion, and G. Varoquaux, “Deriving reproducible biomarkers from multi-site resting-state data: An Autism-based example,” *NeuroImage*, vol. 147, pp. 736–745, Feb. 2017, ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2016.10.045. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1053811916305924> (visited on 03/26/2018).
- [20] P. Bellec, C. Chu, F. Chouinard-Decorte, Y. Benhajali, D. S. Margulies, and R. C. Craddock, “The Neuro Bureau ADHD-200 Preprocessed repository,” *NeuroImage*, Data Sharing Part II, vol. 144, pp. 275–286, Jan. 2017, ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2016.06.034. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S105381191630283X> (visited on 03/15/2018).
- [21] M. R. G. Brown, G. S. Sidhu, R. Greiner, N. Asgarian, M. Bastani, P. H. Silverstone, A. J. Greenshaw, and S. M. Dursun, “ADHD-200 Global Competition: Diagnosing ADHD using personal characteristic data can outperform resting state fMRI measurements,” *Frontiers in Systems Neuroscience*, vol. 6, Sep. 2012, ISSN: 1662-5137. DOI: 10.3389/fnsys.2012.00069. [Online]. Available:

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3460316/> (visited on 03/15/2018).

- [22] C. Minas and G. Montana, “Distance-based analysis of variance: Approximate inference and an application to genome-wide association studies,” *ArXiv:1205.2417 [stat]*, May 2012, arXiv: 1205.2417. [Online]. Available: <http://arxiv.org/abs/1205.2417> (visited on 04/05/2018).
- [23] Y. Li, J. Yosinski, J. Clune, H. Lipson, and J. Hopcroft, “Convergent Learning: Do different neural networks learn the same representations?” *ArXiv:1511.07543 [cs]*, Nov. 2015, arXiv: 1511.07543. [Online]. Available: <http://arxiv.org/abs/1511.07543> (visited on 03/08/2018).
- [24] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *ArXiv:1412.6980 [cs]*, Dec. 2014, arXiv: 1412.6980. [Online]. Available: <http://arxiv.org/abs/1412.6980> (visited on 04/06/2018).