ABSTRACT

Title of Dissertation:	GEOMETRIC AND TOPOLOGICAL RECONSTRUCTION
By:	Michael G. Rawson Doctor of Philosophy, 2022
Dissertation directed by:	Professor Radu Balan Professor Thomas Ernst Professor Michael Robinson

The understanding of mathematical signals is responsible for the information age. Computation, communication, and storage by computers all use signals, either implicitly or explicitly, and use mathematics to manipulate those signals. Reconstruction of a particular signal can be desirable or even necessary depending on how the signal manifests and is measured. We explore how to use mathematical ideas to manipulate and represent signals. Given measurements or samples or data, we analyze how to produce, or *reconstruct*, the desired signal and the fundamental limits in doing so. We focus on reconstruction through a geometric and topological lens so that we can leverage geometric and topological constraints to solve the problems. As inaccuracies and noise are present in every computation, we adopt a statistical outlook and prove results with high probability given noise. We start off with probability and statistics and then use that for active reconstruction where the probability signal needs to be estimated statistically from sampling various sources. We prove optimal ways to doing this even in the most challenging of situations. Then we discuss functional analysis and how to reconstruct sparse rank one decompositions of operators. We prove optimality of certain matrix classes, based on geometry, and compute the worst case via

sampling distributions. With the mathematical tools of functional analysis, we introduce the optimal transportation problem. Then we can use the Wasserstein metric and its geometry to provably reconstruct sparse signals with added noise. We devise an algorithm to solve this optimization problem and confirm its ability on both simulated data and real data. Heavily under-sampled data can be ill-posed which is often the case with magnetic resonance imaging data. We leverage the geometry of the motion correction problem to devise an appropriate approximation with a bound. Then we implement and confirm in simulation and on real data. Topology constraints are often present in non-obvious ways but can often be detected with persistent homology. We introduce the barcode algorithm and devise a method to parallelize it to allow analyzing large datasets. We prove the parallelization speedup and use it for natural language processing. We use topology constraints to reconstruct word-sense signals. Persistent homology is dependent on the data manifold, if it exists. And it is dependent on the manifold's reach. We calculate manifold reach and prove the instability of the formulation. We introduce the combinatorial reach to generalize reach and we prove the combinatorial reach is stable. We confirm this in simulation. Unfortunately, reach and persistent homology are not an invariant of hypergraphs. We discuss hypergraphs and how they can partially reconstruct joint distributions. We define a hypergraph and prove its ability to distinguish certain joint distributions. We give an approximation and prove its convergence. Then we confirm our results in simulation and prove its usefulness on a real dataset.

GEOMETRIC AND TOPOLOGICAL RECONSTRUCTION

by

Michael G. Rawson

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2022.

Advisory Committee:
Professor Radu Balan, Advisor, Chair
Professor Thomas Ernst, Co-Advisor
Associate Professor Michael Robinson, Co-Advisor
Professor Wojciech Czaja
Associate Professor Maria Cameron
Associate Professor Behtash Babadi

Acknowledgments

Acknowledgments are necessary for the many people that made this work possible. My first course at the University of Maryland at College Park was with Prof. John Benedetto. It is always a shock coming into a new environment and Prof. Benedetto gave me reassurance and confidence that guided me throughout my PhD degree which I am very thankful for. But even before my first course started, I met with Prof. Radu Balan. I was surprised by Prof. Balan's generosity to give me time even before I officially joined the university. Prof. Balan guided me on choosing problems, starting them, finishing them, and publishing. I could not have made these accomplishments without this crucial guidance. After arriving at the university, I attended a talk by Prof. Robert Ghrist about Applied Topology which sounded like an oxymoron to me. It turned out his mentee, Prof. Michael Robinson, wrote the textbook [92] on it and was just down the street. How lucky can you be. So I took an idea to Prof. Michael Robinson and we started working on it. My first idea was terrible, but I am glad Prof. Robinson did not tell me and let me figure it out and learn the meta-lesson about how to avoid ideas, at least some, that will be terrible. Prof. Robinson helped me write, communicate, calculate, and publish, of which I am very thankful. In the last two years, I was able to apply my new knowledge in the university's School of Medicine under Prof. Thomas Ernst. I learned that it takes time to learn a new field which was MRI and MRI physics. Prof. Ernst demonstrated great patience in me which I hope to emulate.

Working with the MRI team, I also learned, in a sense, how physicists and engineers think and work. This is very useful for interdisciplinary work and I learned to think in a practical, testable, and adaptive way. Finally, I would like to acknowledge Prof. Wojciech Czaja, Prof. Maria Cameron, and Prof. Behtash Babadi. In their classes, RITs, and seminars, I have been exposed to many areas and concepts unknown to me and their masterful lectures have enlightened me.

Table of Contents

Chapter 1

Introduction

Mathematical reconstructions use information or data to construct a complete structure useful to the application at hand. Information or data can be defined as numbers or a sequence of numbers. First, we'll talk about numbers and how they can be defined. Then, we'll talk about more abstract generalizations. We will not define here everything that is used in this thesis, but we demonstrate how layer by layer, everything can be defined exactly.

The tools of mathematics are numbers. The natural numbers are those you can count on your fingers, 1, 2, 3, and so on. Counting to the next number is the same as adding 1, or at least we can define addition to do this. We will assume the first number, 1, exists. This is our first axiom. Define the successor operation, S, to take any natural number and produce it's natural number successor. We say that the natural numbers, $\mathbb{N}$, are closed under S since for any n, $S(n)$ is a natural number. Our next axiom says for any natural numbers n and m, $S(n) = S(m)$ if and only if $n = m$. With this axiom, we call S an *injection*. We need a first number, so by axiom, there is no natural number n such that $S(n) = 1$. Finally, we need to limit the natural numbers to all of the successors following 1. In other words, for every

natural number n, $S(S(S(...(S(1))...))) = n$ for enough applications of S. With this setup, we can *define* 2 as $S(1)$ and 3 as $S(S(1))$ and so on. Now from this we can define addition recursively. Define $n+1$ to be $S(n)$, that is $n+1 := S(n)$. Define $n + S(m) := S(n+m)$. Using this definition,

$$n + S(m) = S(n+m) = S(S(n+...)) = ... = S^m(n+1) = S^m(S(n)) = S^m(S^n(1))$$

where S^k is just k applications of S. Throughout, we have been using the common notions of logic and equality. Defining the natural numbers and addition from logic was the work of Peano, Peirce, Dedekind, and others in the late 1800s [41, 75, 76, 90].

From the natural numbers, $\mathbb{N}$, we will define the integers, $\mathbb{Z}$. To do this, we will use equivalence classes of pairs of natural numbers denoted (a, b) for a, b natural numbers. Define the integers as the following equivalence classes on all pairs of natural numbers, $\mathbb{N} \times \mathbb{N}$. Define two pairs of natural numbers, (a, b) and (c, d), to be equivalence, $(a, b) \sim (c, d)$, when $a + d = c + b$. We represent the equivalence class of a pair (a, b) with brackets, $[(a, b)]$. For example, $[(2, 1)] = [(3, 2)]$ since $2 + 2 = 3 + 1$ and more generally $[(a, b)] = [(a + 1, b + 1)]$. So this *defines* $0 := [(1, 1)]$ and $-1 := [(1, 2)]$ where 1 and 2 are natural numbers. This also *defines* the *integer* $1 := [(2, 1)]$ with the *natural numbers* 2 and 1. Now we need to define addition on the *integers*. We can do this by setting

$$[(a, b)] + [(c, d)] := [(a + c, b + d)]$$

where the right hand side is adding natural numbers defined previously. Finally we have integers and addition, but for convenience we write an integer k instead of $[(k + 1, 1)]$ and $-k$ instead of $[(1, k + 1)]$. As for subtraction, we define $a - b := a + (-b)$ for any $a, b \in \mathbb{Z}$ and $-c := -[(d, e)] = [(e, d)]$ for any $c \in \mathbb{Z}$ and corresponding representation $d, e \in \mathbb{N}$. The

way we have defined addition and subtraction, they are between just two numbers, not 3 or 4 different numbers. Define adding many numbers by starting with the left two and adding that to the next left most number and so on. Define subtracting many numbers as just adding many negative numbers [15].

Let us define negative numbers. A negative number is any $n \in \mathbb{Z}$ where $n < 0$. Now we need to define the operator $<$. We define $a < b$ true for $a, b \in \mathbb{N}$ when $a + c = b$ for some $c \in \mathbb{N}$. Now on the integers, we define $[(a, b)] < [(c, d)]$ for $a, b, c, d \in \mathbb{N}$ when $a + d < c + b$. We almost have our inequalities except for the reverse. Define, for any $a, b \in \mathbb{N}$ or $a, b \in \mathbb{Z}$, $a > b$ true when $b < a$ true [15].

In a similar way, we can define rational numbers, $\mathbb{Q}$. We define rational numbers as the equivalence classes of pairs of integers where for $(p, q) \sim (r, s)$ when $ps = rq$ for any $p, q, r, s \in \mathbb{Z}$. We need to mention that multiplication of $a, b \in \mathbb{N}$ is $ab := a + a\ S(b)$ and $a1 := a$. Our definition of multiplication on $\mathbb{Z}$ uses this as follows. For $a, b, c, d \in \mathbb{N}$, the product $[(a, b)][(c, d)] := [(ac + bd, ad + bc)]$ defines multiplication of integers. Now for convenience, we do not write rational numbers as $[(p, q)]$, we write $\frac{p}{q}$ with the understanding that this is one valid representation out of many, for example, $\frac{2p}{2q}$ is the *same* rational number [12].

Rational numbers form the number line which can be used to measure any length with arbitrarily high precision. However, the rational numbers cannot solve many standard equations. For example, the length of the hypotenuse of a right triangle with unit (equal 1) legs is not a rational number. We shall again turn to equivalence classes to define irrational numbers using rational numbers. Define the irrational numbers as equivalence classes of limits of sequences of rational numbers. Write an infinite sequence of rational numbers as

$\{p_i\}_{i=1}^{\infty}$ where $p_i \in \mathbb{Q}$ for each $i \in \mathbb{N}$. Now we say the sequence *converges* to a limit $L \in \mathbb{Q}$ if for every $\epsilon > 0, \epsilon \in \mathbb{Q}$ there exists a $M \in \mathbb{N}$ such that for every $m > M$ with $m \in \mathbb{N}$, $|p_m - L| < \epsilon$. Here we used absolute value defined as $|x| := x$ if $x \geq 0$ and $|x| := -x$ if $x < 0$. Some sequences of rational numbers do *not* converge to any rational number, for example $\{p_i\}_{i=1}^{\infty}$ where $p_i := \frac{p_{i-1}}{2} + \frac{1}{p_{i-1}}$ and $p_1 = 2$. This limit $L \in \mathbb{Q}$, if it exists, has $L = L/2 + 1/L$ or $L^2 = LL = 2$ [94].

We say two sequences, $\{p_i\}_{i=1}^{\infty}, \{q_i\}_{i=1}^{\infty}$, converge together when for every $\epsilon > 0, \epsilon \in \mathbb{Q}$ there exists a $M \in \mathbb{N}$ such that for every $m, n > M$ with $m, n \in \mathbb{N}$, $|p_m - q_n| < \epsilon$. Back to our definition of irrational numbers, each irrational number is defined as an equivalence class of sequences of rational numbers. Let rational sequence $\{p_i\}_{i=1}^{\infty} \sim \{q_i\}_{i=1}^{\infty}$ when they converge together. The constant rational sequence $\{\frac{a}{b}\}_{i=1}^{\infty}$ defines an equivalence class for each rational number $\frac{a}{b}$. Putting all of these equivalence classes together, we get the rational and irrational numbers. We henceforth define the Real numbers, $\mathbb{R}$, as all of the equivalence classes include rational and irrational numbers. The number line for $\mathbb{R}$ looks the same as the number line for $\mathbb{Q}$ however $\mathbb{R}$ also contains all of the tiny holes or limits missing in $\mathbb{Q}$. For notational convenience we write the real number $\frac{p}{q}$ the same way though it represents the equivalence class of the constant sequence [94].

Unfortunately, the real numbers, $\mathbb{R}$, still cannot solve many equations. What real number r has $r^2 = -1$? For any $r \in \mathbb{R}$, $r^2 \geq 0$ and $0 > -1$ so $r^2 > -1$. Then $r^2 \neq -1$. When defining real numbers, we defined $\sqrt{2}$ to be a real number and now we will define $\sqrt{-1}$ to be a number. We will call $\sqrt{-1}$ a complex number instead of a real number for historical reasons. While anointing $\sqrt{-1}$ is arbitrary, $\sqrt{-1}$ can solve any equation of the form $r^k = -m$ by $r = m^{1/k}(\sqrt{-1})^{2/k}$. For convenience we will write $i := \sqrt{-1}$ though we will not always

use this definition. We define the complex numbers, $\mathbb{C}$, to contain i and $\mathbb{R}$ and be closed under multiplication and addition. So for any $r, k \in \mathbb{R}$, $r + ki \in \mathbb{C}$. Note that the addition and multiplication in $r + ki$ are *formal* that is just a method of writing equivalent to writing (r, k). By associativity and commutativity of addition and multiplication, any $z \in \mathbb{C}$ can be written as $r + ki$ with some $r, k \in \mathbb{R}$ [95].

So far we have given many categories, memberships, and classes. Another term for this is a *set*, instantiated with $\{\}$. Sets have many useful operations defined: subset $\subset$, element $\in$, union $\cup$, intersection $\cap$, difference $\setminus$, and product $\times$. We use the set theory framework to give operations to our groups, now sets, of numbers, $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$, and $\mathbb{R}$. Specifically we use the Zermelo-Fraenkel Axioms or ZFC, see [32, 64] for all of the details. For example, one may view the *set* $\mathbb{N}$ as a subset of *set* $\mathbb{Z}$, written $\mathbb{N} \subset \mathbb{Z}$. Indeed, we can call any mathematical object a *set*. Even 1 is a set. The most important set however is the empty set written $\emptyset$ which equals the intersection of two disjoint sets, $A \cap B = \emptyset$.

Once we define the power set, $\mathcal{P}(X)$ of a set X, we have a *topology*. Define the power set, $\mathcal{P}(X)$, to be the set that contains, specifically, each set that is a subset of X. So $A \in \mathcal{P}(X)$ if and only if $A \subset X$. Now a topology, τ, of a set X is a subset of the power set, $\mathcal{P}(X)$ such that:

1. $\emptyset \in \tau$ and $X \in \tau$,

2. any union of elements in τ is also in τ, and

3. any intersection of finitely many elements of τ is also in τ.

The largest (or finest) possible τ is when $\tau = \mathcal{P}(X)$ and the smallest (or coarsest) possible τ is when $\tau = \{\emptyset, X\}$. The standard topology τ on the set $\mathbb{R}$ is that containing every open

interval $(a, c) := \{b \in \mathbb{R} : a < b < c\}$ for all $a, c \in \mathbb{R}$ and their unions and finite intersections. We call a subset $U \subset X$ *open* when $U \in \tau$ for X. And we call U *closed* when $\bar{U} := X \backslash U \in \tau$, but it can happen that U is open *and* closed [70].

The way we have defined a topology, τ, for a set X, $X \subset \cup_\alpha U_\alpha$ where $\{U_\alpha\}$ are all of the sets in τ. We say that a set $\{U_\alpha\}$ *covers* X when $X \subset \cup_\alpha U_\alpha$. And in the case that, for each α, $U_\alpha \in \tau$ or U_α is open, we call the set $\{U_\alpha\}$ an *open cover* [70].

As soon as we introduced sets, we saw that $\mathbb{N} \subset \mathbb{Z}$ and then we partitioned $\mathbb{R}$ into many intervals. Considering many different sets, we get a large tree with edges (or relations) where a set is a subset of another set. Key here is the transitive feature of subset where if set $A \subset B$ and set $B \subset C$ then set $A \subset C$. S. Eilenberg and S. Mac Lane generalized this idea by defining *categories* to follow these rules and represent any sets or objects and their relations [64]. A *category* C is

1. a set of objects, $ob(C)$,
2. a set of relations between objects, $hom(C)$, called *morphisms* and
3. a binary operation, $\circ$, on $hom(b, c) \times hom(a, b)$ to $hom(a, c)$ for any $a, b, c \in ob(C)$, called composition.

The binary operation must also follow

1. associativity where $(f \circ g) \circ h = f \circ (g \circ h)$ for any $a, b, c, d \in ob(C)$ and $h \in hom(a, b), g \in hom(b, c), h \in hom(c, d)$ and
2. identity where for any $a, b \in ob(C)$ there exist the identities $1_a \in hom(a, a)$ and $1_b \in hom(b, b)$ and then $1_b \circ f = f = f \circ 1_a$ for any $f \in hom(a, b)$.

Now back to the example of sets, we have the sets be $ob(C)$ and we have a $f \in hom(X, Y)$ for sets X and Y when $X \subset Y$. The binary composition takes a $X \subset Y$ and $Y \subset Z$ and produces $X \subset Z$ for such sets X, Y, Z. Associativity holds because $W \subset X \subset Y \subset Z$ is the unique morphism $W \subset Z$, for such sets W, X, Y, Z. The identity composition holds because $X \subset X \subset Y$ is morphism $X \subset Y$ and $X \subset Y \subset Y$ is also morphism $X \subset Y$, for such sets X, Y [64].

Consider another category, C, formed with the integers, $\mathbb{Z}$. Let $\mathbb{Z} = ob(C)$ and let $f \in hom(n, m)$ when n and m are even (equal to $2k$ for some $k \in \mathbb{Z}$). This is a category as one can check, but is there a way to transform this while maintaining the category properties? Indeed there is and we will show that by moving the morphisms from the evens to the odds (that is $\mathbb{Z} \backslash \{even\ numbers\}$). We will denote F to do the work. Let F take every morphism and shift it up by 1. So for every $f \in hom(2k, 2w)$, $F(f) \in hom(2k + 1, 2w + 1)$. In this example F does not change $ob(C)$ but that need not always be the case so we will specify here that for every $n \in \mathbb{Z}$, $F(n) = n$. Call the new category D. We write $F : C \to D$ and call it a *functor*. A functor must map each object in $ob(C)$ to some object in $ob(D)$ and each morphism in $hom(C)$ to some morphism in $hom(D)$ where

1. for every $a \in ob(C)$, $F(1_a) = 1_{F(a)}$, called identity, and

2. for every $f \in hom(a, b), g \in hom(b, c)$, $F(g \circ f) = F(g) \circ F(f)$, called *functorality*.

Let's consider another type of set, G, where the set G has defined a function called '+' that takes $a, b \in G$ and produces some $c \in G$ written $c := a + b \in G$. Our sets $\mathbb{N}, \mathbb{Z}, \mathbb{R}$ all satisfy this using their '+' but we want to add a requirement so that for each $a \in G$ there is a $b \in G$ where *identity* $0 = a + b \in G$. We call the set G a *group* [6] if

1. associative, $a + (b + c) = (a + b) + c$ for any $a, b, c \in G$,

2. identity, there is an identity $e \in G$ where $e + a = a = a + e$ for any $a \in G$, and

3. inverse, for every $a \in G$ there is a unique $b \in G$ where identity $e = a + b$.

Then $\mathbb{N}$ is not a group because it does not have negative numbers. However, $\mathbb{Z}, \mathbb{Q}$, and $\mathbb{R}$ are groups with '+' written $(\mathbb{Z}, +), (\mathbb{Q}, +)$, and $(\mathbb{R}, +)$. Now the reals, $\mathbb{R}$, using product, $\cdot$, instead of addition, $+$, also satisfies the rules of a group but the identity element changes from 0 to 1. We can write the smallest group by $\{0\}$ where $0 + 0 = 0$. This is the smallest group regardless of which *symbol* is used to describe it's element and we denote the group Z_1. The next largest group can be written as $\{0, 1\}$ where $0+0 = 0, 0+1 = 1$, and $1+1 = 0$. The statement $1 + 1 = 0$ is wrong for integers but here we define it to be true for the *symbols* 0 and 1 in our set. We call this group Z_2. We can keep going and define any group $Z_n = \{0, 1, ..., n-1\}$ by defining $a +_Z b := (a+b)\%n$ for $a, b \in Z_n$. We use $\%$ as the modulo operator where $a\%n := b$ where $0 \leq b = a - kn$ for the largest such $k \in \mathbb{Z}$ [6]. Now that we have covered some of the basics tools, we will continue with the thesis.

These basic tools are crucial for geometric and topological reconstruction. Later, we will often start with some data or measurements. These measurements are in some space such as $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$, or $\mathbb{C}$ which we defined above. Then we calculate geometry, topology, missing data, etc. The topology rules above, and their generalization to category theory are used to construct many of the methods and proofs. Implicitly, the group theory and abstract algebra above is used when we handle rings, vector spaces, and function spaces. However, there are many concepts that we have not covered but will use later and it is up to the reader to self-educate those topics.

1.1 Thesis Contributions

This thesis makes various contributions.

Section 2.1.1 proves Deep Epsilon Greedy Method convergence [82]. We prove that Epsilon Greedy method regret upper bound is minimized with cubic root exploration. We performs experiments with the real-world dataset MNIST, and witness how with either high or low noise, some methods do and some do not converge which agrees with our proof of convergence.

Section 2.1.2 also introduces generalizations of linear upper confidence bound methods which we call DeepUCB. We give the analytic analysis and description. Empirical results show that the Deep UCB often outperforms state of the art methods [83].

Chapter 3 gives new solutions and proofs to types of Feichtinger problems [9]. We perform numerical simulations to confirm our results.

Section 4.2 introduces optimal transport and entropy based super resolution [84]. We prove its convergence, robustness, and stability. We perform numerical simulations and real data calculations to confirm our results and show the usefulness in applications.

Chapter 5 introduces a new analysis and algorithm called MGRAPPA [89]. We give the convergence proof. We perform numerical simulations and real data experiments to confirm the analysis and show usefulness in applications.

Section 6.1 details how to implement a parallel barcode algorithm on CPU or GPU [86]. We analyze the computational complexity and reductions analytically. We perform numerical simulations to confirm our analysis. Then, in Section 6.2, persistent homology is applied to natural language processing to calculate word senses in order to show the

usefulness in applications [85].

Chapter 7 shows instability in Federer's reach formulation in theory and practice [88]. In Section 7.1, we introduce the combinatorial reach to generalize the reach concept. We prove convergence and stability. We demonstrate the usefulness on various geometries.

Chapter 8 defines an entropic hypergraph from a collection of measurements [87]. We prove the necessity of this hypergraph construction to identify many distributions. We introduce an algorithm to approximate the hypergraph and prove the convergence properties. We perform numerical simulations and real data analysis to confirm our result and show usefulness in applications.

Chapter 2

Statistical Signal Reconstruction

Probability is a subject with a long history that goes back as far as gambling. Probability is pattern recognition or signal recognition in a nondeterministic environment. Once the correct probabilities are know, an accurate prediction can be made. Knowing the true probabilities in reality is impossible. Thus comes the invention of *statistics*. Statistics tries to approximate probabilities by collecting data or samples. Let's say that you sample or measure the temperature in many places in a room and always get a different number. Statistics is the intuition that if you measure again, you will probably get a number close to the average of the collected samples. Later on, the information age began with the mass production of computers. Quickly, statisticians began collecting and storing as much data as possible on computers and then began automating the statistical computations. A brand new discipline, with its own questions and solutions, grew up combining statistics and computer science, called *machine learning*.

Machine learning is the study of machines that study. More concretely, it is about algorithms that adapt to data. By this description, statistical estimators are part of machine

learning if performed by a machine. Avoiding the issue of whether humans are a type of machine, we will focus on modern computers and the algorithms that they can compute or run. Since we must limit datasets to finite size, what is the best way to fill in gaps in datasets? As we eluded, the average of the data points is one such estimate. Consider a sequence of points in $\mathbb{R}$ taken across time, the most intuitive method to fill the gaps over time is using a straight line between adjacent data points over time. Now a line segment may contain infinitely many pairs of points which cannot be stored in a computer, but an algorithm can calculate any interim point from storing the line segment end points. This prediction algorithm adapts to the dataset which makes it a classic example of machine learning. In general, creating lines or curves from datapoints is called *interpolation*. In higher dimensions, with a dataset in some $\mathbb{R}^d$, interpolation can calculate a function from $\mathbb{R}^d$ to $\mathbb{R}^k$. If the predictor or dataset comes from the same function calculated from the interpolant, then the prediction will be accurate.

Consider the simple case where the datapoints $(x, y) \in \mathbb{R}^2$ follow $ax + b = y$ for some $a, b \in \mathbb{R}$. To calculate or interpolate a, b we need at least two samples, say (x_1, y_1) and (x_2, y_2). Then solving $\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$ for a, b gives the solution. Given more than two data points, a choice must be made. If all of the data is accurate, all choices give the same solution for a, b, but if some datapoints are inaccurate, the choice matters. Writing all

of the possible equations together for n datapoints yields

$$\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \\ x_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \tag{2.1}$$

which has no solution with inaccurate samples (x_i, y_i). When the errors in (x_i, y_i) are small Equation (2.1) is *approximately* true. Following this notion, we will calculate a, b to make the difference as small as possible,

$$\min_{a,b\in\mathbb{R}} \left\| \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \\ x_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \right\|_2 . \tag{2.2}$$

This is a consistent approximation because as the errors go to 0, the minimizing pair $(\tilde{a}, \tilde{b})$ converges to the true (a, b). This robust estimation method can extend to higher dimensions with more parameters such as $(a, b, c, ...)$. More generally, for any predictors in matrix $X \in \mathbb{R}^{m\times n}$ and responses in vector $y \in \mathbb{R}^n$, minimizing $\|Xv - y\|_2$ over vectors $v \in \mathbb{R}^m$ is known as *linear regression* [107].

Computationally, minimizing $\|Xv-y\|_2$ is nontrivial. The first step is to instead minimize $\|Xv - y\|_2^2 = \langle Xv - y, Xv - y\rangle$. Then

$$\begin{aligned} \langle Xv - y, Xv - y\rangle &= \langle Xv, Xv - y\rangle - \langle y, Xv - y\rangle \\ &= \langle Xv, Xv\rangle - \langle Xv, y\rangle - \langle y, Xv\rangle + \langle y, y\rangle \\ &= v^T X^T Xv - 2v^T X^T y + y^T y. \end{aligned}$$

To optimize, we differentiate with respect to v and set to 0,

$$0 = 2X^T Xv - 2X^T y$$

then

$$v = (X^T X)^{-1} X^T y.$$

Now this local optimizer might be a maximizer or minimizer or saddle or some combination. However, since the objective (2.1) is continuous, differentiable, and convex, this v is the global minimizer [13].

Most interesting processes are not linear. And so linear regression cannot give an accurate estimate in those cases. For datapoints (x, y) where $y = |x|$, how could linear interpolations be combined to estimate such a process? We will need to combine them in a nonlinear way. How would you do so just using the function $ReLu(z) = \mathbb{1}_{z \geq 0} \cdot z$? One answer is

$$ReLu(x \cdot (1, -1)) \cdot (1, 1)^T = y$$

where function $ReLu$ is applied entrywise. This simple example generalizes and a sequence of repeated linear maps and $ReLu$ functions can approximate any Lipschitz continuous function [99]! This model is known as a neural network [107]. As we saw with linear regression, when errors are present in samples (x_i, y_i), there may be no neural network parameters $\theta \in \mathbb{R}^k$ where the neural network with parameters θ produces y, written as $\Phi_\theta(x) = y$. Using the same logic from above, we instead minimize the norm of the difference, $\min_\theta \|\Phi_\theta(x) - y\|_2$. This objective is nonconvex and in general has many minimizers and maximizers so differentiating and setting to zero does not give a global minimizer unfortunately. The next simplest approximate solution is starting from a random initial condition and employing gradient descent which will converge to a local minimizer under certain assumptions [113].

2.1 Reinforcement Learning Application

Reinforcement learning is a subfield of machine learning. We described machine learning as machines that adapt to the data. Reinforcement learning adds the requirement that there is a state of a system, decisions, and feedback. Reinforcement happens when the machine or algorithm makes decisions and receives feedback. The algorithm's goal is to reach some goal state for the system. Consider a board game like Chess. The state of the system is the position of each piece. The decision is which piece to move where. The feedback is the score and the goal state is checkmate. Created in the 1400's, 600 years later in 1997, an algorithm beat the reigning Chess World Champion Garry Kasparov using reinforcement learning [51].

A rule that produces a decision from the system state is known as a *policy*. We can write a policy, π, as a function that takes a state $s \in S$ and decides upon an action $a \in A$, $\pi : S \to A$. The optimal policy is the policy that gets the best feedback. Let the feedback be a real number, $\mathbb{R}$, that we call a reward, written $\mathcal{R}(s, a)$ for $s \in S$ and $a \in A$. In some problems or games, for some $s \in S, a \in A$, $\mathcal{R}(s, a)$ may be undefined. So the optimal policy, written π^*, has $\pi^*(s) = \arg\max_{a \in A : \mathcal{R}(s,a) \in \mathbb{R}} \mathcal{R}(s, a)$. This is a simple computation, but in many cases, $\mathcal{R}(s, a)$ cannot be know until the action is chosen which causes the state to change. The simplest solution is to collect reward data in a training phase by running the system with the random policy which simply chooses each action randomly with equal chance. Then at the end of training, if $\mathcal{R}(s, a)$ is known for each valid $(s, a) \in S \times A$, set $\pi(s) = \arg\max_{a \in A : \mathcal{R}(s,a) \in \mathbb{R}} \mathcal{R}(s, a)$, for all details see [36].

Unfortunately, this solution will not solve games like Chess. The problem is that choosing to take the opponents largest piece each time may yet lead to defeat. If the opponent follows

a fixed strategy, then to win, if possible, the entire sequence of actions must be optimized. The optimal policy, starting at state s_1, is

$$\pi^* = \arg\max_{\pi} \sum_{i=1}^{M} \mathcal{R}(\gamma_\pi^{i-1}(s_1), \pi(\gamma_\pi^{i-1}(s_1))) \tag{2.3}$$

where $\gamma_\pi(s)$ gives the state that follows from action $\pi(s)$, written $\gamma_\pi(s) = s'$ and γ^0 is identity. We are assuming that the problem or game finishes within M step, but if the problem or game can continue forever, replace M with ∞. Computationally, solving (2.3) is more difficult. Even after collecting reward data with a training phase and the random policy, producing the optimal policy takes on the order of $|A|^M$ computations. Furthermore, the training phase is longer, on the order of $|A|^M$ steps [36].

Computing all $|A|^M$ combinations is redundant if a state can be reached with more than one sequence of actions or path in state space. A more efficient solution is to store reward data, $\mathcal{R}(s, a)$, in a matrix $R \in \mathbb{R}^{|S|\times|A|}$. During the training phase, check which matrix entries are empty and only choose actions that produce $\mathcal{R}(s, a)$ where matrix entry $R_{s,a}$ is empty. This reduces excess computation but another order $|A|^M$ computation is required to compute π^* of Equation (2.3). Let's setup another matrix called $Q \in \mathbb{R}^{|S|\times|A|}$ to provide the reward of (s, a) *plus* future rewards from the optimal policy. Then

$$Q_{s,a} = \mathcal{R}(s, a) + \max_{a\in A} Q_{\Gamma(s,a),a}$$

where Γ is the function that takes a state and action, (s, a), and produces the next state of the system, $\Gamma(s, a) \in S$. Now to compute this Q matrix during training, still every non-redundant path in state space needs to be traversed, but Q can be computed and updated quickly during training. Once we have Q, then we directly have $\pi^*(s) = \arg\max_{a\in A:Q_{s,a}\in\mathbb{R}} Q_{s,a}$. This reinforcement learning method is known as *Q-learning*. So far we have not allowed for

samples of $\mathcal{R}(s,a)$ to contains errors. Some amount of errors in matrix Q are inevitable and can lead to the wrong policy based on Q, written π_Q. To minimize the effect of errors, common practice is to discount the future with a parameter $0 < \beta \leq 1$ and define

$$Q_{s,a} = \mathcal{R}(s,a) + \beta \max_{a \in A} Q_{\Gamma(s,a),a}.$$

Then

$$\pi_Q = \arg\max_{\pi} \sum_{i=1}^{M} \beta^{i-1} \mathcal{R}(\gamma_\pi^{i-1}(s_1), \pi(\gamma_\pi^{i-1}(s_1)))$$

Of course when the problem or game continues forever, there is no hope of collecting all of the training data in finite time. By setting $\beta < 1$, then for some M large enough, β^M is bounded low enough and if $\mathcal{R}$ is bounded, this proxy problem can be solved.

In some problems or games, a training phase is not allowed. For example, you walk into a casino and you want the optimal policy for what to play and how to play it. But playing for free and 'training' yourself is not allowed. What is the best method to produce the best policy when a training phase is not allowed? Note, no information about the problem or game is known. Reward data from playing is the only data permitted. In the simple case of just $K \in \mathbb{N}$ slot machines, a simple solution is the *Epsilon Greedy Method* in Algorithm 1 from [8]. This method covers problems or games with fixed actions, only one state, $S = \{0\}$, and rewards sampled from fixed distributions. These types of problems are also known as *Multi-Armed Bandits*.

Algorithm 1: Epsilon Greedy Method

Parameters: $K > 1$, $c > 0$, $0 < d < 1$.
Initialization: $\epsilon_n := \min\{1, \frac{cK}{d^2 n}\}$ for $n = 1, 2, ...$
for $n = 1, 2, ...$ **do**
  i_n = the action with the highest current average reward
  if $\eta > 1 - \epsilon_n : \eta \sim Uniform([0, 1])$ **then**
    play i_n
  else
    play a uniform random action
  end
end

The Epsilon Greedy Method does not know the optimal policy initially, but it converges to the optimal policy as time $T \to \infty$, see below. Let $\mu_i := \mathbb{E}(P_i)$ and $\Delta_i := \mu^* - \mu_i$ where * is an optimal action index.

Theorem 1 (Auer 2002 [8]). *For all $K > 1$, and for all reward distributions $P_1, ..., P_K$ with support in $[0, 1]$, if policy Epsilon Greedy Method is run with input parameter $0 < d \leq \min_{i:\mu_i<\mu_*} \Delta_i$, then the probability that after any number $n \geq cK/d$ of plays Epsilon Greedy Method chooses a suboptimal machine j is at most*

$$\frac{c}{d^2 n} + 2\left(\frac{c}{d^2}\log\frac{(n-1)d^2e^{1/2}}{cK}\right)\left(\frac{cK}{(n-1)d^2e^{1/2}}\right)^{c/(5d^2)} + \frac{4e}{d^2}\left(\frac{cK}{(n-1)d^2e^{1/2}}\right)^{c/2}.$$

Remark. *For c large enough (e.g. $c > 5$) the above bound is of order $c/(d^2 n) + o(1/n)$ for $n \to \infty$, as the second and third terms in the bound are $O(1/n^{1+\epsilon})$ for some $\epsilon > 0$ (recall that $0 < d < 1$). Summing over N time steps gives a regret of $O(\log(N))$. The average regret $O(\log(N)/N) \to 0$ as $N \to \infty$ so this converges to an optimal policy.*

There are many other methods with various characteristics and convergence properties. [4, 8].

Now when the state/context may change, the problem is more challenging. In many scenarios, it is assumed that the state is a random variable sampled at each time step and

independent of past actions and rewards. This is known as *unconfoundedness* [7]. In [7, 116], convergence is shown for the Doubly Robust method. When state X is sampled on a low dimensional manifold, fast convergence is shown in [20]. Other neural network based policy learning methods also converge [83, 118]. We will show convergence for Deep Epsilon Greedy method under reasonable assumptions and then discuss variations and generalizations. The convergence of policy learning depends on the policy and the neural network. The neural network's convergence that we use goes back to [45] which bounds the neural network's parameter weights. This is equivalent to a Lipschitz bound on the employed class of neural networks [121].

First we describe the well known Epsilon Greedy method in Algorithm 2. We will use and analyze this algorithm throughout this section. This method runs for M time steps and at each time step takes in a state vector, X_t, and chooses an action, D_t, from $A = \{a_1, ..., a_K\}$. The reward at each time step is recorded and the attempt is to maximize the total rewards received.

Algorithm 2: Deep Epsilon Greedy

Input:
$M \in \mathbb{N}$: Total time steps
$m \in \mathbb{N}$: Context dimension
$X \in \mathbb{R}^{M \times m}$ where state $X_t \in \mathbb{R}^m$ for time step t
$A = \{a_1, ..., a_K\}$: Available Actions
$\Phi : \mathbb{R}^m \to \mathbb{R}$: Untrained Neural Network
$Reward : \mathbb{N}_{[1,K]} \to \mathbb{R}$
Output:
$D \in \mathbb{N}^M$: Decision Record
$R \in \mathbb{R}^M$ where R_t stores the reward from time step t
Begin:
for *t = 1, 2, ..., M* **do**
  for *j = 1 ... K* **do**
    $\hat{\mu}_{a_j} = \Phi_{j,t}(X_t)$ (predict reward)
  end
  $\eta \sim$ Uniform(0,1)
  $\epsilon_t = 1/t$
  if $\eta > \epsilon_t$ **then**
    $D_t = \arg\max_{1 \le j \le K} \ \hat{\mu}_{a_j}$
  else
    $\rho \sim$ Uniform({1,...,K})
    $D_t = A_\rho$
  end
  $R_t = Reward(D_t)$
  (Training Stage)
  for *j = 1 ... K* **do**
    $S_j = \{l : 1 \le l \le t, \ D_l = j\}$
    TrainNNet($\Phi_{j,t-1}, input = X_{S_j}, output = R_{S_j}$)
  end
end

2.1.1 Deep Epsilon Greedy Method Convergence

We will utilize the following theorem.

Theorem 2. *[[45] Theorem 16.3] Let Φ_n be a neural network with n parameters and the parameters are optimized to minimize MSE of the training data, $S = \{(X_i, Y_i)\}$ where X and Y are almost surely bounded. Let the training data, be size n, and $Y_i = R(x_i) \sim N(\mu_{x_i}, \sigma_{x_i})$*

where $R : \mathbb{R}^m \to \mathbb{R}$. Then for n large enough,

$$\mathbb{E}_S \int |\Phi_n(x) - \mathbb{E}(R(x))|^2 dP(x) \leq c\sqrt{\frac{\log(n)}{n}} \tag{2.4}$$

for some $c > 0$.

Now, assume there are K actions to play. Let $T_j(t)$ be the random variable equal to number of times action j is chosen in the first $t-1$ steps. Let $T_j^R(t)$ be the number of times action j is chosen in the first $t-1$ steps by the uniform random branch of the algorithm. Let X be the state vector at some time step t and Y_t^j be the reward of action j at time step t both almost surely bounded. Let $\mu_j(X) := \mathbb{E}(Y_t^j|X)$. We will use $*$ for an optimal action index, for example let $\mu_*(X)$ be the expectation of all optimal actions at X. Let $\Delta_j(X) := \max\{0, \mu_*(X) - \mu_j(X)\}$. Let $\epsilon_t = 1/t$. Let I_t be the action chosen at time t. Assume state X is sampled from an unknown distribution i.i.d. at each time step t.

Theorem 3. *[82] Assume there is optimality gap δ with $0 < \delta \leq \Delta_j(X)$ for all j and X where j is suboptimal. Assume there is at least one suboptimal action for any context. With the assumptions from above and from Theorem 2, the Deep Epsilon Greedy method converges with expected regret approaching 0 almost surely. Let C_i be the constant from Theorem 2 for neural network i and let n_i be the minimal value of the training data size such that Equation (2.4) holds. Set $C_0 = 8\sqrt{2}\max_i C_i$ and $t_0 = \exp(2K\max\{e, \max_i n_i\})$. Then for every $t > t_0$ with probability at least $1 - K\exp(-3\log(t)/(28K))$,*

$$\delta/(tK) \leq \mathbb{E}_{X_t}\mathbb{E}_{I_t}\mathbb{E}_R\left[R_*(X_t) - R(X_t)\right] \leq \frac{\max_i \mathbb{E}_{X_t}\Delta_i(X_t)}{t} + K^{3/2}\frac{C_0}{\delta}\sqrt{\frac{\log(\log(t)) - \log(2K)}{\log(t)}}. \tag{2.5}$$

The expectations in above equations refer to the specific time step t. The probability refers to the stochastic policy's choices at previous time steps, 1 to $t-1$.

Now we consider a more general *rational function* for ϵ_t.

Theorem 4. *[82] Let $\epsilon_t = 1/t^p$ where $0 < p < 1$. Assume the assumptions of Theorem 3. Set $C_0' = 8\sqrt{2(1-p)}\max_i C_i$ and $t_0 > (2(1-p)K\max\{e, \max_i n_i\})^{1/(1-p)}$. Then for every $t > t_0$ with probability at least $1 - K\exp\left(-(3\ t^{-p+1})/(28(-p+1)K)\right)$,*

$$\delta/(Kt^p) \leq \mathbb{E}_{X_t}\mathbb{E}_{I_t}\mathbb{E}_R\left[R_*(X_t) - R(X_t)\right] \tag{2.6}$$

$$\leq \frac{\max_i \mathbb{E}_{X_t}\Delta_i(X_t)}{t^p} + K^{3/2}\frac{C_0'}{\delta}\sqrt{\frac{\log(t^{-p+1}) - \log(2(-p+1)K)}{t^{-p+1}}}. \tag{2.7}$$

The expectations in above equations refer to the specific time step t. The probability refers to the stochastic policy's choices at previous time steps, 1 to $t-1$.

We will need the following lemma.

Lemma 5. *[82] Recall that $T_j^R(t)$ is the number of times action j is chosen in the first $t-1$ steps by the uniform random branch of the algorithm. For the case $\epsilon_t = 1/t$,*

$$\mathbb{P}\left(\bigwedge_{i=1}^{K}\{T_i^R(t) \geq \log(t)/(2K)\}\right) \geq 1 - K\exp\left(-\frac{3\log(t)}{28K}\right). \tag{2.8}$$

For the case $\epsilon_t = 1/t^p$, where $0 < p < 1$,

$$\mathbb{P}\left(\bigwedge_{i=1}^{K}\{T_i^R(t) \geq \frac{t^{-p+1}}{2(-p+1)K}\}\right) \geq 1 - K\exp\left(-\frac{3\ t^{-p+1}}{28(-p+1)K}\right). \tag{2.9}$$

Proof of Lemma 5. Fix i. Recall $\epsilon_t = 1/t$. Following the proof of theorem 3 in [8],

$$\begin{aligned}\mathbb{E}(T_i^R(t)) &= \sum_{l=1}^{t-1}\mathbb{P}(\eta < \epsilon_l \wedge \rho = i) = \sum_{l=1}^{t-1}\mathbb{P}(\eta < \epsilon_l)\mathbb{P}(\rho = i)\\ &= \sum_{l=1}^{t-1}\epsilon_l/K = \frac{1}{K}\sum_{l=1}^{t-1}\frac{1}{l} \geq \frac{1}{K}\log(t)\end{aligned}$$

Set $B(t) := \frac{1}{K}\sum_{l=1}^{t-1}\frac{1}{l}$. Then we have

$$Var(T_i^R(t)) = \sum_{l=1}^{t-1}\frac{\epsilon_l}{K}(1 - \frac{\epsilon_l}{K}) \leq \frac{1}{K}\sum_{l=1}^{t-1}\epsilon_l = \frac{1}{K}\sum_{l=1}^{t-1}\frac{1}{l} = B(t).$$

By Bernstein's inequality

$$\begin{aligned}\mathbb{P}(T_i^R(t) \le B(t)/2) &= \mathbb{P}\left(T_i^R(t) - B(t) \le -B(t)/2\right) \\ &\le \exp\left(\frac{-B(t)^2/8}{Var(T_i^R(t)) + \frac{1}{3}B(t)/2}\right) \\ &\le \exp\left(\frac{-B(t)^2/8}{B(t) + \frac{1}{3}B(t)/2}\right) \\ &\le \exp\left(-\frac{3B(t)}{28}\right) \le \exp\left(-\frac{3\log(t)}{28K}\right).\end{aligned}$$

So by union bound

$$\mathbb{P}\left(\bigvee_{i=1}^{K}\{T_i^R(t) \le \log(t)/(2K)\}\right) \le K\mathbb{P}\left(T_1^R(t) \le \log(t)/(2K)\right) \le K\exp\left(-\frac{3\log(t)}{28K}\right).$$

And

$$\mathbb{P}\left(\bigwedge_{i=1}^{K}\{T_i^R(t) \ge \log(t)/(2K)\}\right) \ge 1 - K\exp\left(-\frac{3\log(t)}{28K}\right).$$

This proves Equation (2.8). Next, we prove Equation (2.9). Fix i. Recall $\epsilon_t = 1/t^p$. Following the proof of theorem 3 in [8],

$$\begin{aligned}\mathbb{E}(T_i^R(t)) &= \sum_{l=1}^{t-1}\mathbb{P}(\eta < \epsilon_l \wedge \rho = i) = \sum_{l=1}^{t-1}\mathbb{P}(\eta < \epsilon_l)\mathbb{P}(\rho = i) \\ &= \sum_{l=1}^{t-1}\epsilon_l/K = \frac{1}{K}\sum_{l=1}^{t-1}\frac{1}{l^p} \ge \frac{1}{(1-p)K}\, t^{1-p}\end{aligned}$$

Set $B(t) := \frac{1}{K}\sum_{l=1}^{t-1}\frac{1}{l^p}$. we have

$$Var(T_i^R(t)) = \sum_{l=1}^{t-1}\frac{\epsilon_l}{K}(1 - \frac{\epsilon_l}{K}) \le \frac{1}{K}\sum_{l=1}^{t-1}\epsilon_l = \frac{1}{K}\sum_{l=1}^{t-1}\frac{1}{l^p} = B(t).$$

By Bernstein's inequality

$$\begin{aligned}\mathbb{P}(T_i^R(t) \leq B(t)/2) &= \mathbb{P}\left(T_i^R(t) - B(t) \leq -B(t)/2\right) \\ &\leq \exp\left(\frac{-B(t)^2/8}{Var(T_i^R(t)) + \frac{1}{3}B(t)/2}\right) \\ &\leq \exp\left(\frac{-B(t)^2/8}{B(t) + \frac{1}{3}B(t)/2}\right) \\ &\leq \exp\left(-\frac{3B(t)}{28}\right) \leq \exp\left(-\frac{3\ t^{-p+1}}{28(-p+1)K}\right).\end{aligned}$$

So by union bound

$$\begin{aligned}\mathbb{P}\left(\bigvee_{i=1}^{K}\{T_i^R(t) \leq \frac{t^{-p+1}}{2(-p+1)K}\}\right) &\leq K\ \mathbb{P}\left(T_1^R(t) \leq \frac{t^{-p+1}}{2(-p+1)K}\right) \\ &\leq K\ \mathbb{P}\left(T_1^R(t) \leq \frac{B(t)}{2}\right) \leq K\exp\left(-\frac{3\ t^{-p+1}}{28(-p+1)K}\right).\end{aligned}$$

And

$$\mathbb{P}\left(\bigwedge_{i=1}^{K}\{T_i^R(t) \geq \frac{t^{-p+1}}{2(-p+1)K}\}\right) \geq 1 - K\exp\left(-\frac{3\ t^{-p+1}}{28(-p+1)K}\right).$$

This prove Equation (2.9). □

Proof of Theorem 3. Let $*$ be an optimal action at X_t and R the reward from the epsilon greedy method. Let $\Phi_{i,t}$ be the trained neural network for action i and have t parameters. By Lemma 5, with probability greater than $1 - K\exp(-3\log(t)/(28K))$, $T_i^R(t) \geq \log(t)/(2K)$ for all i. In this case, we have

$$\begin{aligned}\mathbb{E}_{X_t}\mathbb{E}_{I_t}\mathbb{E}_R[R_*(X_t) - R(X_t)] &= \mathbb{E}_{X_t}[\mu_*(X_t) - \mathbb{E}_{I_t}\mathbb{E}_R R(X_t)] \\ &= \mathbb{E}_{X_t}[\mu_*(X_t) - \sum_{i=1}^{K} \mu_i(X_t)\mathbb{P}(I_t = i|X_t)] \\ &= \mathbb{E}_{X_t} \sum_i \Delta_i(X_t)\mathbb{P}(I_t = i|X_t) \\ &= \sum_i \mathbb{E}_{X_t}\Delta_i(X_t)\mathbb{P}(I_t = i|X_t)\end{aligned}$$

Then

$$\mathbb{E}_{X_t}[\Delta_i(X_t)\mathbb{P}(I_t = i|X_t)] \leq \mathbb{E}_{X_t}\Delta_i(X_t)[\epsilon_t/K + \mathbb{P}(\Phi_{i,t}(X_t) \geq \Phi_{*,t}(X_t))]$$

and, with Markov's inequality,

$$\begin{aligned}\mathbb{P}(\Phi_{i,t}(X_t) \geq \Phi_{*,t}(X_t)) &\leq \mathbb{P}(\Phi_{i,t}(X_t) \geq \mu_i(X_t) + \Delta_i(X_t)/2) + \mathbb{P}(\Phi_{*,t}(X_t) \leq \mu_*(X_t) - \Delta_i(X_t)/2) \\ &= \int_{\mathbb{1}\{\Phi_{i,t}(X_t) \geq \mu_i(X_t) + \Delta_i(X_t)/2\}} dP_i + \int_{\mathbb{1}\{\Phi_{*,t}(X_t) \leq \mu_*(X_t) - \Delta_i(X_t)/2\}} dP_* \\ &\leq \int_{\mathbb{1}\{|\Phi_{i,t}(X_t) - \mu_i(X_t)| \geq \Delta_i(X_t)/2\}} dP_i + \int_{\mathbb{1}\{|\Phi_{*,t}(X_t) - \mu_*(X_t)| \geq \Delta_i(X_t)/2\}} dP_* \\ &\leq \int \frac{|\Phi_{i,t}(X_t) - \mu_i(X_t)|^2}{\Delta_i(X_t)^2/4} dP_i + \int \frac{|\Phi_{*,t}(X_t) - \mu_*(X_t)|^2}{\Delta_i(X_t)^2/4} dP_*\end{aligned}$$

Then

$$\begin{aligned}&\mathbb{E}_{X_t}[\Delta_i(X_t)\mathbb{P}(I_t = i|X_t)] \\ &\leq \mathbb{E}_{X_t}\Delta_i(X_t)\epsilon_t/K + \mathbb{E}_{X_t}\Delta_i(X_t) \int \frac{|\Phi_{i,t}(X_t) - \mu_i(X_t)|}{\Delta_i(X_t)/2} dP_i + \mathbb{E}_{X_t}\Delta_i(X_t) \int \frac{|\Phi_{*,t}(X_t) - \mu_*(X_t)|}{\Delta_i(X_t)/2} dP_* \\ &\leq \mathbb{E}_{X_t}\Delta_i(X_t)\epsilon_t/K + \frac{4}{\delta} \int_{x_t : i \neq *} \int |\Phi_{i,t}(x_t) - \mu_i(x_t)|^2 dP_i dP_{x_t} + \frac{4}{\delta} \int_{x_t : i \neq *} \int |\Phi_{*,t}(x_t) - \mu_*(x_t)|^2 dP_* dP_{x_t},\end{aligned}$$

by dominated convergence,

$$\leq \mathbb{E}_{X_t}\Delta_i(X_t)\ \epsilon_t/K + \frac{4}{\delta} \int \int_{x_t} |\Phi_{i,t}(x_t) - \mu_i(x_t)|^2 dP_{x_t} dP_i + \frac{4}{\delta} \int \int_{x_t} |\Phi_{*,t}(x_t) - \mu_*(x_t)|^2 dP_{x_t} dP_*$$

Recall that we are in the case that $T_i^R(t) \geq \log(t)/(2K)$ for all i. Let C_i be the constant from Theorem 2 for neural network i and let n_i be the minimal value of the training data size such that Equation (2.4) holds. Choose $t_0 > \exp(2K \max\{e, \max_i n_i\})$. Since the map $x \mapsto \sqrt{\frac{ln(x)}{x}}$ is monotone decreasing for $x > e$, the above expression is further upper bounded by

$$\begin{aligned}
&\leq \mathbb{E}_{X_t}\Delta_i(X_t)\ \frac{\epsilon_t}{K} + \frac{4}{\delta}C_i\sqrt{\frac{\log(T_i(t))}{T_i(t)}} + \frac{4}{\delta}C_*\sqrt{\frac{\log(T_*(t))}{T_*(t)}} \\
&\leq \mathbb{E}_{X_t}\Delta_i(X_t)\ \frac{\epsilon_t}{K} + \frac{4C_i}{\delta}\sqrt{\frac{\log(T_i^R(t))}{T_i^R(t)}} + \frac{4C_*}{\delta}\sqrt{\frac{\log(T_*^R(t))}{T_*^R(t)}} \\
&\leq \frac{\mathbb{E}_{X_t}\Delta_i(X_t)}{tK} + \left[\frac{4C_i}{\delta} + \frac{4C_*}{\delta}\right]\sqrt{\frac{\log(\log(t)/(2K))}{\log(t)/(2K)}}
\end{aligned}$$

So

$$\begin{aligned}
&\mathbb{E}_{X_t}\mathbb{E}_{I_t}\mathbb{E}_R[R_*(X_t) - R(X_t)] \\
&= \mathbb{E}_{X_t}[\mu_*(X_t) - \mathbb{E}_{I_t}\mathbb{E}_R R(X_t)] \\
&= \sum_i \mathbb{E}_{X_t}\Delta_i(X_t)\mathbb{P}(I_t = i|X_t) \\
&\leq \frac{\max_i \mathbb{E}_{X_t}\Delta_i(X_t)}{t} + K^{3/2}\sqrt{2}\left[\frac{4\max_i C_i}{\delta} + \frac{4C_*}{\delta}\right]\sqrt{\frac{\log(\log(t)) - \log(2K)}{\log(t)}}
\end{aligned}$$

from where the upper bound in (2.5) follows. To prove the lower bound, we have, for i not optimal, that

$$\begin{aligned}
\mathbb{E}_{X_t}[\Delta_i(X_t)\mathbb{P}(I_t = i|X_t)] &\geq \mathbb{E}_{X_t}\Delta_i(X_t)\epsilon_t/K \\
&\geq \mathbb{E}_{X_t}\delta\epsilon_t/K \geq \delta/(tK).
\end{aligned}$$

Then using the suboptimal action, assumed to exist, we get

$$\mathbb{E}_{X_t}\mathbb{E}_{I_t}\mathbb{E}_R[R_*(X_t) - R(X_t)]$$
$$= \sum_i \mathbb{E}_{X_t}\Delta_i(X_t)\mathbb{P}(I_t = i|X_t) \geq \delta/(tK).$$

□

Corollary 5.1. *[82] The Epsilon Greedy method with any predictor, neural network or otherwise, with convergence of $c\sqrt{\frac{\log(n)}{n}}$, or better, will have regret converging to 0 almost surely.*

Remark. *[82] With $\epsilon_t = 1/t^p$ with $p \leq 1$, enough samples will be taken to train an approximation to convergence. When $p > 1$, The number of samples is finite and the approximation will not converge in general. This is called a starvation scenario since the optimal action is not sampled sufficiently.*

Corollary 5.2. *[82] The optimal p for $\epsilon_t = 1/t^p$ with the fastest converging upper bound of Theorem 3 for Deep Epsilon Greedy is $p = 1/3$.*

Proof of Corollary 5.2. From the above remark, we know $p \leq 1$. First we show that $0 < p < 1$ converges faster than $p = 1$. Theorem 3 gives the bound of

$$\frac{\max_i \mathbb{E}_{X_t}\Delta_i(X_t)}{t} + K^{3/2}\frac{C_0}{\delta}\sqrt{\frac{\log(\log(t)) - \log(2K)}{\log(t)}}$$

for $p = 1$. Theorem 4 gives the bound of

$$\frac{\max_i \mathbb{E}_{X_t}\Delta_i(X_t)}{t^p} + K^{3/2}\frac{C_0'}{\delta}\sqrt{\frac{\log(t^{-p+1}) - \log(2(-p+1)K)}{t^{-p+1}}} \tag{2.10}$$

for $p < 1$. Set $\alpha_t := \max_i \mathbb{E}_{X_t}\Delta_i(X_t)$

and $\beta_t = K^{3/2}\frac{C_0}{\delta}\sqrt{\log(\log(t)) - \log(2K)}$

and $\gamma_t = K^{3/2}\frac{C_0'}{\delta}\sqrt{\log(t^{-p+1}) - \log(2(1-p)K)}$. Using the ratio test, the limit of the ratio of the bounds is

$$\begin{aligned}
&\lim_{t\to\infty} \frac{\alpha_t t^{-p} + \gamma_t t^{(p-1)/2}}{\alpha_t t^{-1} + \beta_t/\sqrt{\log(t)}} \\
&= \frac{\lim_{t\to\infty} \alpha_t t^{-p}\beta_t^{-1}\sqrt{\log(t)} + \gamma_t\beta_t^{-1}t^{(p-1)/2}\sqrt{\log(t)}}{\lim_{t\to\infty} \alpha_t t^{-1}\beta_t^{-1}\sqrt{\log(t)} + 1} \\
&= \lim_{t\to\infty} t^{(p-1)/2}\sqrt{\log(t)} = 0
\end{aligned}$$

for $0 < p < 1$. Now we find p to minimize Equation (2.10).

$$\begin{aligned}
&\lim_{t\to\infty} \frac{\max_i \mathbb{E}_{X_t}\Delta_i(X_t)}{t^p} + K^{3/2}\frac{C_0}{\delta}\sqrt{\frac{\log(t^{-p+1}) - \log(2(1-p)K)}{t^{-p+1}}} \\
&= t^{-p}\lim_{t\to\infty}\alpha_t + K^{3/2}\frac{C_0}{\delta}\sqrt{\frac{\log(t^{-p+1}) - \log(2(1-p)K)}{t^{-3p+1}}}
\end{aligned}$$

The convergence rate is t^{-p} if $-3p + 1 \geq 0$. Otherwise, $p > 1/3$, the rate achieved is $t^{-(1-p)/2} > t^{-(1-1/3)/2} = t^{-1/3}$. So the optimal is at $p = 1/3$.

□

Proof of Theorem 4. Let $*$ be an optimal action at X_t and R the reward from the epsilon greedy method. Let $\Phi_{i,t}$ be the trained neural network for action i and have t parameters. By Lemma 5, with probability greater than $1 - K\exp\left(-\frac{3\ t^{-p+1}}{28(-p+1)K}\right)$, $T_i^R(t) \geq \frac{t^{-p+1}}{2(-p+1)K}$ for all i. In this case, we have

$$
\begin{aligned}
&\mathbb{E}_{X_t}\mathbb{E}_{I_t}\mathbb{E}_R[R_*(X_t) - R(X_t)] \\
&\quad = \mathbb{E}_{X_t}[\mu_*(X_t) - \mathbb{E}_{I_t}\mathbb{E}_R R(X_t)] \\
&\quad = \mathbb{E}_{X_t}[\mu_*(X_t) - \sum_{i=1}^{K} \mu_i(X_t)\mathbb{P}(I_t = i|X_t)] \\
&\quad = \mathbb{E}_{X_t} \sum_i \Delta_i(X_t)\mathbb{P}(I_t = i|X_t) \\
&\quad = \sum_i \mathbb{E}_{X_t}\Delta_i(X_t)\mathbb{P}(I_t = i|X_t)
\end{aligned}
$$

Then

$$
\mathbb{E}_{X_t}[\Delta_i(X_t)\mathbb{P}(I_t = i|X_t)] \leq \mathbb{E}_{X_t}\Delta_i(X_t)[\epsilon_t/K + \mathbb{P}(\Phi_{i,t}(X_t) \geq \Phi_{*,t}(X_t))]
$$

and, with Markov's inequality,

$$
\begin{aligned}
\mathbb{P}(\Phi_{i,t}(X_t) \geq \Phi_{*,t}(X_t)) &\leq \mathbb{P}(\Phi_{i,t}(X_t) \geq \mu_i(X_t) + \Delta_i(X_t)/2) + \mathbb{P}(\Phi_{*,t}(X_t) \leq \mu_*(X_t) - \Delta_i(X_t)/2) \\
&= \int_{\mathbb{1}\{\Phi_{i,t}(X_t) \geq \mu_i(X_t) + \Delta_i(X_t)/2\}} dP_i + \int_{\mathbb{1}\{\Phi_{*,t}(X_t) \leq \mu_*(X_t) - \Delta_i(X_t)/2\}} dP_* \\
&\leq \int_{\mathbb{1}\{|\Phi_{i,t}(X_t) - \mu_i(X_t)| \geq \Delta_i(X_t)/2\}} dP_i + \int_{\mathbb{1}\{|\Phi_{*,t}(X_t) - \mu_*(X_t)| \geq \Delta_i(X_t)/2\}} dP_* \\
&\leq \int \frac{|\Phi_{i,t}(X_t) - \mu_i(X_t)|^2}{\Delta_i(X_t)^2/4} dP_i + \int \frac{|\Phi_{*,t}(X_t) - \mu_*(X_t)|^2}{\Delta_i(X_t)^2/4} dP_*
\end{aligned}
$$

Then

$$
\begin{aligned}
&\mathbb{E}_{X_t}[\Delta_i(X_t)\mathbb{P}(I_t = i|X_t)] \\
&\leq \mathbb{E}_{X_t}\Delta_i(X_t)\epsilon_t/K + \mathbb{E}_{X_t}\Delta_i(X_t) \int \frac{|\Phi_{i,t}(X_t) - \mu_i(X_t)|}{\Delta_i(X_t)/2} dP_i + \mathbb{E}_{X_t}\Delta_i(X_t) \int \frac{|\Phi_{*,t}(X_t) - \mu_*(X_t)|}{\Delta_i(X_t)/2} dP_*
\end{aligned}
$$

$$
\leq \mathbb{E}_{X_t}\Delta_i(X_t)\epsilon_t/K + \frac{4}{\delta} \int_{x_t : i \neq *} \int |\Phi_{i,t}(x_t) - \mu_i(x_t)|^2 dP_i dP_{x_t} + \frac{4}{\delta} \int_{x_t : i \neq *} \int |\Phi_{*,t}(x_t) - \mu_*(x_t)|^2 dP_* dP_{x_t},
$$

by dominated convergence,

$$\leq \mathbb{E}_{X_t}\Delta_i(X_t)\ \epsilon_t/K + \frac{4}{\delta}\int\int_{x_t} |\Phi_{i,t}(x_t) - \mu_i(x_t)|^2 dP_{x_t} dP_i + \frac{4}{\delta}\int\int_{x_t} |\Phi_{*,t}(x_t) - \mu_*(x_t)|^2 dP_{x_t} dP_*$$

Recall that we are in the case that $T_i^R(t) \geq \frac{t^{-p+1}}{2(-p+1)K}$ for all i. Let C_i be the constant from Theorem 2 for neural network i and let n_i be the minimal value of the training data size such that Equation (2.4) holds. Choose $t_0 > (2(1-p)K\max\{e, \max_i n_i\})^{1/(1-p)}$. Since the map $x \mapsto \sqrt{\frac{ln(x)}{x}}$ is monotone decreasing for $x > e$, the above expression is further upper bounded by

$$\begin{aligned}
&\leq \mathbb{E}_{X_t}\Delta_i(X_t)\ \frac{\epsilon_t}{K} + \frac{4}{\delta}C_i\sqrt{\frac{\log(T_i(t))}{T_i(t)}} + \frac{4}{\delta}C_*\sqrt{\frac{\log(T_*(t))}{T_*(t)}} \\
&\leq \mathbb{E}_{X_t}\Delta_i(X_t)\ \frac{\epsilon_t}{K} + \frac{4C_i}{\delta}\sqrt{\frac{\log(T_i^R(t))}{T_i^R(t)}} + \frac{4C_*}{\delta}\sqrt{\frac{\log(T_*^R(t))}{T_*^R(t)}} \\
&\leq \frac{\mathbb{E}_{X_t}\Delta_i(X_t)}{Kt^p} + \left[\frac{4C_i}{\delta} + \frac{4C_*}{\delta}\right]\sqrt{\frac{\log(t^{-p+1}/(2(-p+1)K))}{t^{-p+1}/(2(-p+1)K)}}
\end{aligned}$$

So

$$\begin{aligned}
&\mathbb{E}_{X_t}\mathbb{E}_{I_t}\mathbb{E}_R[R_*(X_t) - R(X_t)] \\
&= \mathbb{E}_{X_t}[\mu_*(X_t) - \mathbb{E}_{I_t}\mathbb{E}_R R(X_t)] \\
&= \sum_i \mathbb{E}_{X_t}\Delta_i(X_t)\mathbb{P}(I_t = i|X_t) \\
&\leq \frac{\max_i \mathbb{E}_{X_t}\Delta_i(X_t)}{t^p} + K^{3/2}\sqrt{2(1-p)}\left[\frac{4\max_i C_i}{\delta} + \frac{4C_*}{\delta}\right]\cdot\sqrt{\frac{\log(t^{-p+1}) - \log(2(-p+1)K)}{t^{-p+1}}}
\end{aligned}$$

from where (2.7) follows. To prove the lower bound, we have, for i not optimal, that

$$\begin{aligned}
\mathbb{E}_{X_t}[\Delta_i(X_t)\mathbb{P}(I_t = i|X_t)] &\geq \mathbb{E}_{X_t}\Delta_i(X_t)\epsilon_t/K \\
&\geq \mathbb{E}_{X_t}\delta\epsilon_t/K \geq \delta/(Kt^p).
\end{aligned}$$

Then using the suboptimal action, assumed to exist, we get

$$\mathbb{E}_{X_t}\mathbb{E}_{I_t}\mathbb{E}_R[R_*(X_t) - R(X_t)] = \sum_i \mathbb{E}_{X_t}\Delta_i(X_t)\mathbb{P}(I_t = i|X_t) \geq \delta/(Kt^p).$$

□

2.1.2 Deep Upper Confidence Bound Method

Deep Upper Confidence Bound (Deep UCB) method [83] for high dimensional, nonlinear CMAB problems is motivated by Upper Confidence Bound (UCB) method [8] for MAB where the arm with the highest expected reward plus uncertainty is chosen. Deep UCB uses two neural networks. One neural network predicts the expected reward and the second neural network predicts uncertainty. The arms with the highest expected rewards plus high uncertainty are chosen. Unlike the other methods, nonlinear reward functions are accurately modeled and exploration is guided to maximize certainty. Choosing uncertain arms decreases the uncertainty. As uncertainty of all arms approaches zero, this method converges to choosing the arm with the highest expected reward.

We model the expectation of reward functions, $\mu_c, c \in \mathcal{C}$, from a neural network where the input is the context vector $c \in \mathcal{C}$ and the output is a vector approximating the reward, $NN : \mathcal{C} \to \mathbb{R}$. Further, we estimate the uncertainty of the expected reward, σ_c, given a context vector $c \in \mathcal{C}$, represented as the standard error of the expectation. This procedure is done for each arm. Then, for each arm, the deep upper confidence bound (DUCB) estimate is defined, given $c \in \mathcal{C}$, as

$$DUCB(c) = \hat{\mu}_c + \hat{\sigma}_c/\sqrt{n}$$

by the arm's n realizations, empiric expectation $\hat{\mu}_c$, and empiric standard deviation $\hat{\sigma}_c$. Here,

μ and σ are unknown functions of the context vector; $\mu, \sigma : \mathcal{C} \to \mathbb{R}$. While we estimate $\hat{\mu}_c$ approximately with $NN_1 = \hat{\mu}$, we approximate σ with another neural network, $NN_2 : \mathcal{C} \to \mathbb{R}$ and $NN_2 = \hat{\sigma}^2$, and thus,

$$DUCB(c) = NN_1(c) + \sqrt{NN_2(c)/n}$$

where n is number of realizations of the arm and c is the context vector. As implemented, the neural networks have multiple outputs and will compute the prediction for each arm simultaneously and then calculation is as above but with vectors.

When training these neural networks, the mean squared error (MSE) of NN_1 is minimized and the L^1 error of NN_2 is minimized. Let $R(c)$ be a realization of the reward of a context vector c for an arm and $S \subset \mathcal{C}$ be some subset of the context vectors. Then, for a sufficiently parameterized NN,

$$NN_1 = \arg\min_{NN_1} \sum_{c \in S} (NN_1(c) - R(c))^2$$

is the maximum likelihood esimator for each c and converges when the reward function $R(c) \sim$ Gaussian. Likewise,

$$NN_2 = \arg\min_{NN_2} \sum_{c \in S} |NN_2(c) - (NN_1(c) - R(c))^2|.$$

For each c, with T independent realizations of R and $t = 1, ..., T$,

$$\sum_{1 \le t \le T} |NN_2(c) - (NN_1(c) - R_t(c))^2|$$

$$= \sum_{t \in T_1} NN_2(c) - (NN_1(c) - R_t(c))^2 - \sum_{t \in T_2} NN_2(c) - (NN_1(c) - R_t(c))^2$$

where $T_1 = \{1 \le t \le T : NN_2(c) - (NN_1(c) - R_t(c))^2 \ge 0\}$ and $T_2 = [1, T] \backslash T_1$. As $NN_1(c) \to \mu_c$,

$$\sum_{t \in T_1} NN_2(c) - (NN_1(c) - R_t(c))^2 \to T_1 \cdot NN_2(c) - \sum_{t \in T_1} (\mu_c - R_t(c))^2$$

Likewise for T_2. By minimizing $\sum_T |NN_2(c) - (NN_1(c) - R(c))^2|$, then $NN_2(c) \to \sigma_c^2$ as $NN_1(c) \to \mu_c$ and $T \to \infty$. Therefore, DUCB converges to the upper confidence bound given sufficiently parameterized neural networks and enough training samples.

We will call this method Deep UCB2, given in Algorithm 3. Note that by assuming the arms are similar stochastic functions (but still unknown), we have simplified Algorithm 3. We call a variation of this method Deep UCB1, given in Algorithm 4.

Algorithm 3: Deep UCB2

$C \in \mathbb{R}^{T \cdot K \cdot m}$ where $C_{t,j}$ is the j^{th} context vector in $\mathbb{R}^m$ at time step t
$A = \{arm_1, ..., arm_N\}$
NN_{1,Z_t}, NN_{2,Z_t} = Neural Network : $\mathbb{R}^m \to \mathbb{R}$ with Z_t neurons
$Reward : \mathbb{N}_{[1,K]} \to \mathbb{R}$
for *t = 1 ... T* **do**
  for *j = 1 ... K* **do**
    # $NN(\cdot)$ gives a prediction using NN
    $\hat{\mathbb{E}}_R(arm_j) = NN_{1,Z_t}(C_{t,j})$ # Predict reward
    $\hat{\mathbb{E}}_V(arm_j) = NN_{2,Z_t}(C_{t,j})$ # Predict certainty
  end
  for *j = 1 ... K* **do**
    # Choose arms
    $D_{t,j} = \arg\max_{arm \in A - D_{t,:}} \hat{\mathbb{E}}_R(arm) + \sqrt{\frac{\hat{\mathbb{E}}_V(arm)}{t}}$
  end
  for *j = 1 ... K* **do**
    $R_{t,j} = Reward(D_{t,j})$ # Record rewards
    $V_{t,j} = (R_{t,j} - \hat{\mathbb{E}}_R(arm_j))^2$ # Record accuracy
  end
  # Train the networks
  TrainNNet($NN_{1,Z_t},\ in = C_{1:t,D_{1:t,1:K}},\ out = R_{1:t,1:K},\ loss = MSE$)
  TrainNNet($NN_{2,Z_t},\ in = C_{1:t,D_{1:t,1:K}},\ out = V_{1:t,1:K},\ loss = L1$)
end

Algorithm 4: Deep UCB1

$C \in \mathbb{R}^{T \cdot K \cdot m}$ where $C_{t,j}$ is the j^{th} context vector in $\mathbb{R}^m$ at time step t
$A = \{arm_1, ..., arm_N\}$
NN_{1,Z_t}, NN_{2,Z_t} = Neural Network : $\mathbb{R}^m \to \mathbb{R}$ with Z_t neurons
$Reward : \mathbb{N}_{[1,K]} \to \mathbb{R}$
$\epsilon_i = \mu_{i,max} - \mu_{i,min} + \epsilon$
for t = *1 ... T* **do**
  if $t \leq JN$ **then**
    # Initial exploration phase
    for j = *1 ... K* **do**
      # Choose arms
      $D_{t,j} = ((\lfloor \frac{t-1}{J} \rfloor + j - 1) \mod N) + 1$
    end
  else
    for j = *1 ... K* **do**
      # $NN(\cdot)$ gives a prediction using NN
      $\hat{\mathbb{E}}_R(arm_j) = \frac{1}{W_t} \sum_{i=1}^{W_t} NN_{R,Z_t}^{(i)}(C_{t,j})$ # Predict reward
      $\hat{\mathbb{E}}_V(arm_j) = \frac{1}{W_t} \sum_{i=1}^{W_t} NN_{V,Z_t}^{(i)}(C_{t,j})$ # Predict certainty
    end
    for j = *1 ... K* **do**
      # Choose arms
      $D_{t,j} = \arg\max_{arm \in A - D_{t,:}} \hat{\mathbb{E}}_R(arm) + \sqrt{\frac{2\hat{\mathbb{E}}_V(arm) + 2\log t}{\sqrt{|\{d \in D_{1:t-1,:} \ s.t. \ d = arm\}|}}} + \epsilon_{arm}$
    end
    for j = *1 ... K* **do**
      $R_{t,j} = Reward(D_{t,j})$ # Record rewards
      $V_{t,j} = (R_{t,j} - \hat{\mathbb{E}}_R(arm_j))^2$ # Record accuracy
    end
    # Train the networks
    for i = *1 ... W_t* **do**
      $J_i = [\![(i-1)\frac{t}{W_t} + 1, i\frac{t}{W_t}]\!]$
      TrainNNet($NN_{R,Z_t}^{(i)}$, $in = C_{J_i, D_{t,1:K}}$, $out = R_{J_i,1:K}$, loss=MSE)
      TrainNNet($NN_{V,Z_t}^{(i)}$, $in = C_{J_i, D_{t,1:K}}$, $out = V_{J_i,1:K}$, loss=MSE)
    end
  end
end

2.1.3 Optical Character Recognition Application

We experiment with the MNIST [57] dataset which contains real world, handwritten digits 0-9. The action set is $\{a_1, ..., a_5\}$. The state, X_t at each time step, t, is 5 random

images. Each digit has an equal chance of being chosen. The reward $Y_t = R(X_t, I_t)$ is the digit of the image corresponding to the chosen action plus Gaussian noise. We plot the regret convergence to 0 of the Deep Epsilon Greedy method in Fig. 2.1. We get a convergence rate of approximately $t^{-1/2}$ which is within the bounds of Theorem 3.

Next, we compare the Uniform Random method (see Algorithm 5), the Linear Regression method (see Algorithm 7), the LinUCB method [62] (see Algorithm 8), the Deep Epsilon Greedy method (see Algorithm 2), and the Simple Deep Epsilon Greedy method (only 1 hidden layer). The Deep Epsilon Greedy method uses 3 convolutional layers followed by a fully connected layer of width 100. The Simple Deep Epsilon Greedy method uses just one fully connected layer of width 100. For all methods, training is every 20 time steps. For the neural networks, number of training epochs is always 16 and the initial learning rate is 10^{-3}. We plot the reward normalized (divided by the time step) in Fig. 2.2. Each curve is an average of 12 independent runs.

We see that in both the high noise and low noise case, Deep Epsilon Greedy converges to the optimal but Simple Deep Epsilon Greedy does not. Simple Deep Epsilon does not have the necessary complexity to converge required by Theorem 2. Because the neural network does not converge, the regret in this policy does not converge to 0. The LinUCB and Linear Regression also cannot converge to the solution because they are linear models but this is a nonlinear problem. So they perform as well as purely random actions.

We have shown convergence guarantees for the Deep Epsilon Greedy method, Algorithm 2. In Corollary 5.1, we have shown convergence of generalizations to other common predictive models. We have shown convergence failure with $\epsilon_t = 1/t^p$ for $p > 1$. In Corollary 5.2, we showed that $\epsilon_t = 1/t^{1/3}$ gives the fastest convergence bound. To see these results

in experiments, we perform a standard MNIST [57] experiment. The converging methods vs non-converging methods is empirically confirmed and displayed in Fig. 2.1 and Fig. 2.2.

2.1.4 Open Problems

- Is there a tighter bound for Theorem 3?
- Is there a function $\epsilon(t)$ with a tighter bound than the bound in Theorem 4?
- Is there a tighter bound for Theorem 2, since that would improve the bound in Theorem 3?

Algorithm 5: Random

Input:
$M \in \mathbb{N}$: Total time steps
$m \in \mathbb{N}$: Context dimension
$X \in \mathbb{R}^{M \times m}$ where state $X_t \in \mathbb{R}^m$ for time step t
$A = \{a_1, ..., a_K\}$: Available Actions
$Reward : \mathbb{N}_{[1,K]} \to \mathbb{R}$
Output:
$D \in \mathbb{N}^M$: Decision Record
$R \in \mathbb{R}^M$ where R_t stores the reward from time step t
Begin:
for *t = 1, 2, ..., M* **do**
    $\rho \sim$ Uniform({1,...,K})
    $D_t = A_\rho$ (Choose Random Action)
    $R_t = Reward(D_t)$
end

Algorithm 6: Optimal

Input:
$M \in \mathbb{N}$: Total time steps
$m \in \mathbb{N}$: Context dimension
$X \in \mathbb{R}^{M \times m}$ where state $X_t \in \mathbb{R}^m$ for time step t
$A = \{a_1, ..., a_K\}$: Available Actions
$Reward : \mathbb{N}_{[1,K]} \to \mathbb{R}$
$oracle : \mathbb{N}_{[1,M]} \to \mathbb{N}_{[1,K]}$: Oracle for correct action index
Output:
$D \in \mathbb{N}^M$: Decision Record
$R \in \mathbb{R}^M$ where R_t stores the reward from time step t
Begin:
for *t = 1, 2, ..., M* **do**
  $D_t = oracle(t)$ (Choose Correct Action)
  $R_t = Reward(D_t)$
end

Algorithm 7: Linear

Input:
$M \in \mathbb{N}$: Total time steps
$m \in \mathbb{N}$: Context dimension
$X \in \mathbb{R}^{M \times m}$ where state $X_t \in \mathbb{R}^m$ for time step t
$A = \{a_1, ..., a_K\}$: Available Actions
$Reward : \mathbb{N}_{[1,K]} \to \mathbb{R}$
Output:
$B_j \in \mathbb{R}^m$: Linear Models for $1 \leq j \leq K$
$D \in \mathbb{N}^M$: Decision Record
$R \in \mathbb{R}^M$ where R_t stores the reward from time step t
Begin:
for *j = 1, 2, ..., K* **do**
  $B_j = 0$
end
for *t = 1, 2, ..., M* **do**
  for *j = 1, 2, ..., K* **do**
    $\hat{\mu}_{a_j} = B_j^T \ X_t$ (predict rewards)
  end
  $D_t = \arg\max_{1 \leq j \leq K} \ \hat{\mu}_{a_j}$
  $R_t = Reward(D_t)$
  (Training Stage)
  $S = \{l : 1 \leq l \leq t, \ D_l = D_t\}$
  $B_{D_t} = \arg\min_B \|R_S - BX_S^T\|_2$
end

Algorithm 8: LinUCB

Input:

$M \in \mathbb{N}$: Total time steps

$m \in \mathbb{N}$: Context dimension

$X \in \mathbb{R}^{M \times m}$ where state $X_t \in \mathbb{R}^m$ for time step t

$A = \{a_1, ..., a_K\}$: Available Actions

$Reward : \mathbb{N}_{[1,K]} \to \mathbb{R}$

Output:

$B_j \in \mathbb{R}^{m \times m}$: Linear Maps for $1 \leq j \leq K$

$b_j \in \mathbb{R}^m$: Linear Models for $1 \leq j \leq K$

$D \in \mathbb{N}^M$: Decision Record

$R \in \mathbb{R}^M$ where R_t stores the reward from time step t

Begin:

for *j = 1, 2, ..., K* **do**

  $B_j = I$

  $b_j = 0$

end

for *t = 1, 2, ..., M* **do**

  for *j = 1, 2, ..., K* **do**

    $\Theta = B_j^{-1} b_j$

    (Predict Rewards)

    $\hat{\mu}_{a_j} = \Theta^T \ X_t + \sqrt{X_t^T B_j^{-1} X_t}$

  end

  $D_t = \arg\max_{1 \leq j \leq K} \ \hat{\mu}_{a_j}$

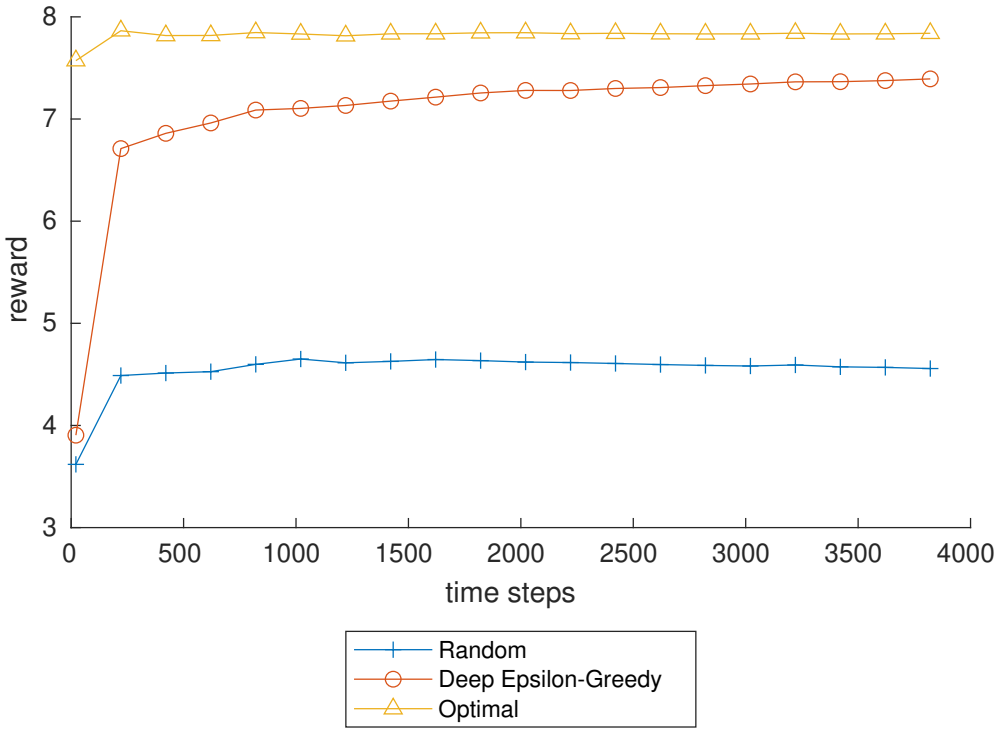
  $R_t = Reward(D_t)$

  (Training Stage)

  $B_{D_t} = B_{D_t} + X_t X_t^T$

  $b_{D_t} = b_{D_t} + R_t X_t$

end



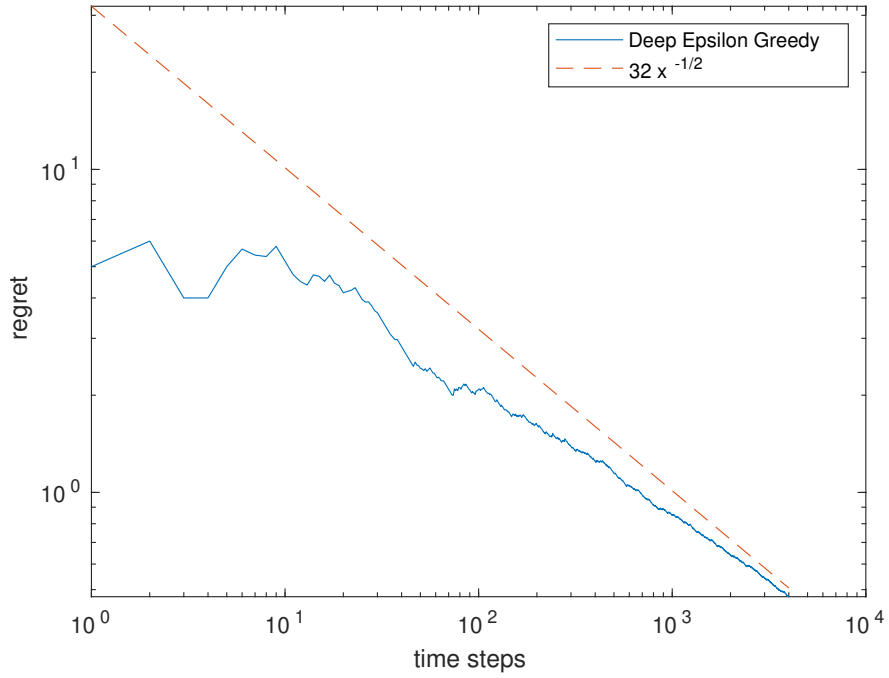


Figure 2.1: Deep Epsilon Greedy method convergence of regret to 0 at rate $x^{-1/2}$. Plotting normalized reward of optimal method minus normalized reward of Deep Epsilon Greedy method. No noise added to MNIST dataset. Single run with 1000 neurons in the fully connected, final layer.

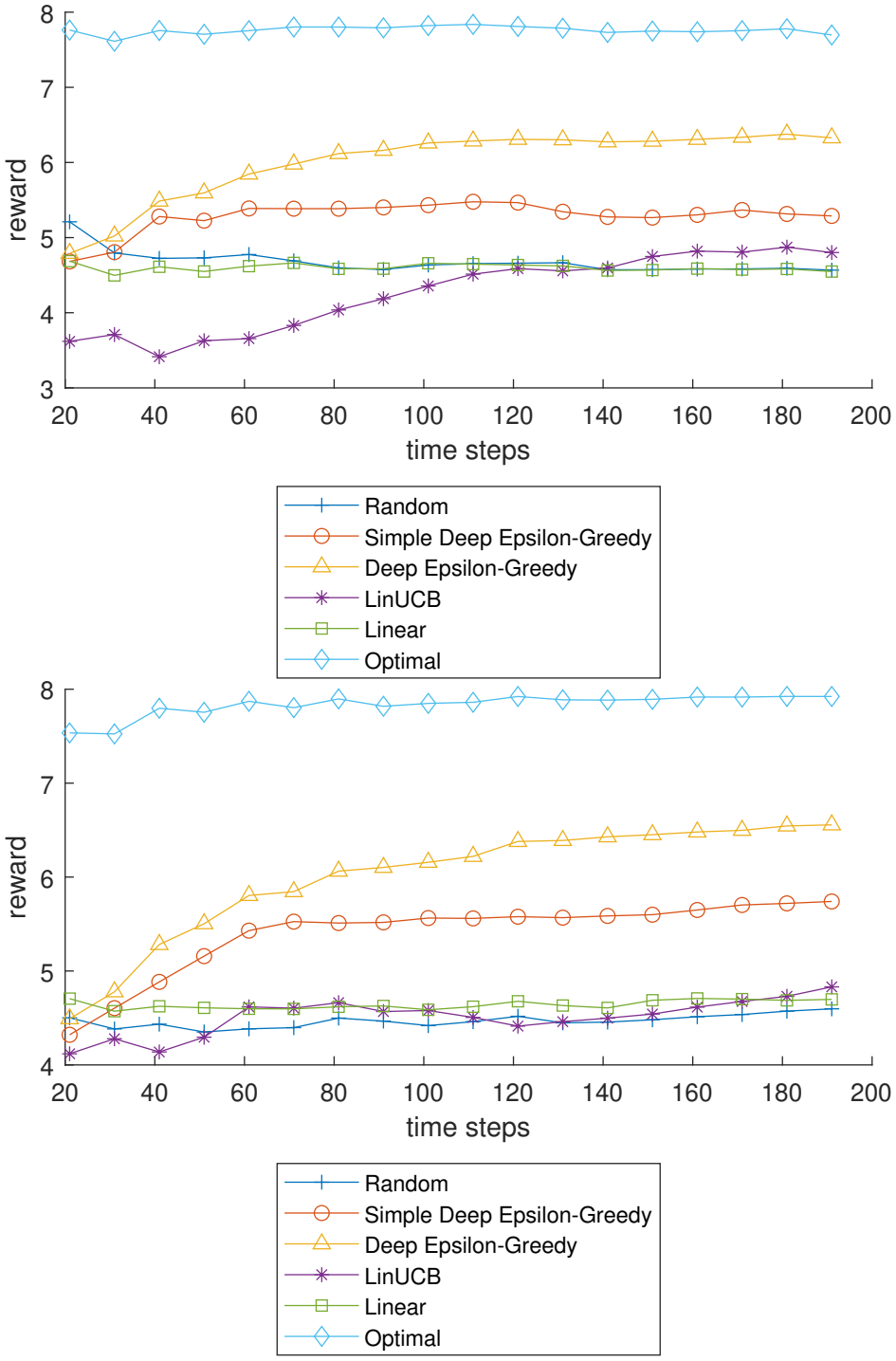


Figure 2.2: Top: Low Noise with no Gaussian noise added to reward. Bottom: High Noise with Gaussian noise, sigma = 1, added to reward. Both: Mean reward normalized (divide by time step) plotted over time steps for each method. Task is to choose the largest MNIST image (digit) of 5 random images. No Gaussian noise added to reward. Mean is over 12 independent runs.

Chapter 3

Functional Analysis Based Reconstruction with Ray Separation Applications

Functional analysis analyzes *functionals*, also known as *operators*, which are maps from a space to a scalar field [96]. In multivariate applications and problems, samples are vectors and together form a matrix. Functional analysis allows classification and decomposition of operators, including matrices. Often multivariate applications inhabit a low dimensional subspace. This corresponds to the *spectrum* of the data matrix being sparse, or the matrix being *low rank* [6, 96]. Reconstructing the low dimensional subspace is a challenging problem in estimation and optimization. We will discuss reconstructing sparse matrix decompositions using the geometry of the L^1 unit ball which naturally increases sparsity.

In signal processing, mixed signals often need to be processed and distilled. If we consider

waves in a medium as signals, reflections in the medium will mix a signal with itself. We will consider mixed signals in the form of an operator which in finite dimensions is a matrix. We distill, or reconstruct, the signals by decomposing the matrix into an integral of rank one matrices. Many different measures will satisfy the decomposition, so we construct an optimization problem to calculate the measure that maximizes sparsity. This can be viewed as the simplest solution of the geometry problem.

Define the l_1 unit sphere $S_1 = \{x \in \mathbb{C}^N : \|x\|_1 = 1\}$. Solve $\mu^* = \arg\min_\mu \int_{S_1} d\mu(x)$ such that $A = \int_{S_1} xx^T d\mu(x)$ where μ is a measure on $\mathbb{C}^N$ and $A \in \mathbb{C}^{N\times N}$ is the measurement operator. For example, μ can be any matrix decomposition such as the eigendecomposition $A = \int_{S_1} xx^T d\mu(x) = \sum_i \lambda_i \|v_i\|_1^2 \frac{v_i}{\|v_i\|_1}\frac{v_i^T}{\|v_i\|_1}$. We conjecture that μ^* is a discrete, finite measure: $\mu^*(x) = \sum_{i=1}^M w_i \delta_{x_i}(x)$ where δ is the Dirac delta measure and $w_i \in \mathbb{C}\ \forall i$. Indeed, we prove that in some situations, this is true.

When restricted to discrete measures, this optimization problem can be formulated as: solve $A = \sum_{n\geq 1} g_n g_n^*$ for vectors $g_n \in \mathbb{C}^N$ and minimize $\sum_{n\geq 1} \|g_n\|_1^2$. Then each rank one matrix corresponds to the autocorrelation of signal g_n. We define the following relaxed inf's.

Definition 3.0.1. For $A \in S_+^n = \{B \in \mathbb{C}^{n\times n} : B = B^*, B \geq 0\}$,

$$\gamma_+(A) := \inf_{A=\sum_{n\geq 1} g_n g_n^*} \sum_{n\geq 1} \|g_n\|_1^2.$$

If $A \in S^n = \{B \in \mathbb{C}^{n\times n} : B = B^*\}$,

$$\gamma(A) := \inf_{A=\sum_{n\geq 1} g_n h_n^*} \sum_{n\geq 1} \|g_n\|_1 \|h_n\|_1$$

and

$$\gamma_0(A) := \inf_{A=B-C; B,C\in S_+^n} (\gamma_+(B) + \gamma_+(C)).$$

We introduce and investigate the properties of various measures of optimality of such decompositions [9]. For some classes of positive semidefinite matrices, we give explicitly these optimal decompositions. These classes include diagonally dominant matrices and certain of their generalizations; 2×2, and a class of 3×3 matrices.

Recall that a matrix $A \in S_+^n(\mathbb{C})$ is said to be diagonally dominant if $A_{ii} \geq \sum_{j=1}^n |A_{ij}|$ for each $i = 1, 2, ..., n$. If the inequality is strict for each i, we say that the matrix is strictly diagonally dominant. The following result applies to any diagonally dominant matrix in S_+^n.

Theorem 6. *[9] Let $A \in S_+^n(\mathbb{C})$ be a diagonally dominant matrix. Then $\gamma(A) = \gamma_0(A) = \gamma_+(A)$.*

Proof. Let $e_i = (0, ..., 0, 1, 0, ..., 0)$ and $u_{i,j}(x) = (0, ..., \sqrt{x}, ..., \overline{\sqrt{x}}, ..., 0)$. Given a diagonally dominant matrix A, we consider the following decomposition of A,

$$A = \sum_{i<j} u_{i,j}(A_{ij}) u_{i,j}(A_{ij})^* + \sum_i (A_{ii} - \sum_{j \in \{1,..,n\} \setminus \{i\}} |A_{ij}) e_i e_i^*.$$

It follows that

$$\begin{aligned}
\gamma_+(A) &\leq \sum_{i<j} 4|A_{ij}| + \sum_i (A_{ii} - \sum_{j \in \{1,..,n\} \setminus \{i\}} |A_{ij}|) \\
&= \sum_{i<j} 4|A_{ij}| + \sum_i A_{ii} - \sum_i \sum_{j \in \{1,..,n\} \setminus \{i\}} |A_{ij}| \\
&= \sum_{i<j} 4|A_{ij}| + \sum_i A_{ii} - \sum_{i<j} 2|A_{ij}| \\
&= \|A\|_{1,1}.
\end{aligned}$$

□

Theorem 7. *[9] Assume $A \in S_+^n$ admits a decomposition*

$$A = \sum_{1 \leq i < j \leq n} u_{ij} u_{ij}^* + \sum_{i=1}^n v_i v_i^*$$

where each u_{ij} has non-zero entries at most on positions i and j, and each v_i has non-zero entries at most on position i. Then $\gamma_+(A) = \|A\|_{1,1}$.

Proof. The hypothesis implies

$$u_{ij} = (0, ..., 0, c_{ij;i}, 0, ..., 0, c_{ij;j}, 0, ..., 0)^T$$

and

$$v_i = (0, ..., 0, d_i, 0, ..., 0)^T$$

where $c_{ij;i}$ is on position i, $c_{ij;j}$ is on position j, and d_i is on position i. Without loss of generality, we can assume $d_i \in \mathbb{R}$ and $c_{ij;i}, c_{ij;j} \in \mathbb{C}$. We write $A = (a_{ij})_{i,j=1}^n$ where for $1 \leq i < j \leq n$, $a_{ij} = c_{ij;i}\overline{c_{ij;j}}$, whereas for $1 \leq i \leq n$,

$$a_{ii} = d_i^2 + \sum_{j=1}^{i-1} |c_{ji;i}|^2 + \sum_{j=i+1}^{n} |c_{ij;i}|^2.$$

These imply

$$\sum_{1\leq i<j\leq n} \|u_{ij}\|_1^2 + \sum_{i=1}^{n} \|v_i\|_1^2 = \sum_{1\leq i<j\leq n} (|u_{ij;i}| + |u_{ij;j}|)^2 + \sum_{i=1}^{n} d_i^2 = \sum_{1\leq i,j\leq n} |a_{i,j}| = \|A\|_{1,1}.$$

□

Theorem 8. *[9] Suppose that $A \in S_+^2$, then*

$$\gamma_+(A) = \|A\|_{1,1}.$$

Proof. If $A = uu^*$ is a rank 1 matrix in S_+^2, the proof is straightforward. Suppose $A \in S_+^2$ is rank 2. $A = \begin{bmatrix} a & c \\ \bar{c} & b \end{bmatrix}$ with $ab - |c|^2 > 0$. Using the Lagrangian decomposition [117] we can write

$$A = \begin{bmatrix} \sqrt{a} \\ \frac{\bar{c}}{\sqrt{a}} \end{bmatrix} \begin{bmatrix} \sqrt{a} & \frac{c}{\sqrt{a}} \end{bmatrix} + \begin{bmatrix} 0 \\ \sqrt{b - \frac{|c|^2}{a}} \end{bmatrix} \begin{bmatrix} 0 & \sqrt{b - \frac{|c|^2}{a}} \end{bmatrix}$$

The result then follows. □

For certain 3×3 matrices the Lagrangian decomposition [117] is optimal. In particular, we have the following result.

Theorem 9. *[9] Let $A \in S_+^3$ be of rank 2 or 3. If*

$$A = \begin{pmatrix} a & b & c \\ \bar{b} & d & e \\ \bar{c} & \bar{e} & f \end{pmatrix}$$

then

$$\gamma_+(A) \leq \|A\|_{1,1} + \frac{2(|ae - \bar{b}c| + |b||c| - a|e|)}{a}.$$

In particular, if $|ae-\bar{b}c|+|b||c| = a|e|$ then $\gamma_+(A) = \|A\|_{1,1}$ and the Lagrangian decomposition (which in this case is the LDL factorization) is optimal.

Proof. We first assume that A has rank 3. In this case, A must be positive definite and $adf \neq 0$. Indeed, if one of the diagonal term, say $f = 0$, then using the fact that $A \in S_+^3$ would implies that $df - |e|^2 = -|e|^2 > 0$ which is impossible. Let

$$u_1 = \tfrac{1}{\sqrt{a}} A\delta_1 = \begin{bmatrix} \sqrt{a} \\ \frac{\bar{b}}{\sqrt{a}} \\ \frac{\bar{c}}{\sqrt{a}} \end{bmatrix},$$

where $\{\delta_i\}_{i=1}^3$ is the standard ONB for $\mathbb{C}^3$. By Theorem 2.10 in [9], the matrix $A - u_1 u_1^*$ is rank 2. In fact, in this case, the rank 2 matrix is given by

$$A - u_1 u_1^* = \begin{bmatrix} 0 & 0 & 0 \\ 0 & d - \frac{|b|^2}{a} & e - \frac{\overline{b}c}{a} \\ 0 & \overline{e} - \frac{\overline{c}b}{a} & f - \frac{|c|^2}{a} \end{bmatrix}$$

Let

$$u_2 = \frac{1}{\sqrt{d - \frac{|b|^2}{a}}}(A - u_1 u_1^*)\delta_2 = \begin{bmatrix} 0 \\ \sqrt{d - \frac{|b|^2}{a}} \\ \frac{\overline{e} - \frac{\overline{c}b}{a}}{\sqrt{d - \frac{|b|^2}{a}}} \end{bmatrix}.$$

It follows that $A - u_1 u_1^* - u_2 u_2^* = u_3 u_3^*$ where

$$u_3 = \begin{bmatrix} 0 \\ 0 \\ \sqrt{\frac{\det A}{ad - |b|^2}} \end{bmatrix}.$$

Consequently, the Lagrange decomposition of A is $A = u_1 u_1^* + u_2 u_2^* + u_3 u_3^*$ which implies that

$$\gamma_+(A) \leq \sum_{k=1}^{3} \|u_k\|_1^2 = \|A\|_{1,1} + \frac{2(|ae - \overline{b}c| + |b||c| - a|e|)}{a}.$$

Now suppose that the rank of A is 2. In this case, it is possible for $adf = 0$. However, only one of the diagonal element can be 0. So assume that $f = 0$, then we also get that $e = c = 0$. In this case

$$A = \begin{bmatrix} a & b & 0 \\ \overline{b} & d & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

which reduces to Theorem 8. Thus, we may assume without loss of generality that $adf \neq 0$. In this case, we can proceed as above. However, because the rank of the matrix A is now 2

we see that $A = u_1u_1^* + u_2u_2^*$ and

$$\gamma_+(A) \leq \|u_1\|_1^2 + \|u_2\|_1^2 = \|A\|_{1,1} + \frac{2(|ae-\bar{b}c|+|b||c|-a|e|)}{a}.$$

□

Remark.

1. *If one of the off diagonal elements b, or c is 0, then Theorem 9 shows that the Lagrange decomposition is optimal for* $\gamma_+(A)$.

2. *Suppose* $n = 4$ *and let* $V = \frac{1}{\sqrt{14}} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & -1 \\ 1 & 1 \end{pmatrix}$, *and consider*

$$A = VV^T = \frac{1}{14} \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & -1 & 1 \\ 1 & -1 & 2 & 0 \\ 1 & 1 & 0 & 2 \end{pmatrix}.$$

Then A has rank 2, and the $\|A\|_{1,1} = 1$. *However,* $\gamma_+(A) \neq \gamma(A)$.

3.1 Statistical Computation

Here we inspect upper bounds of $\gamma_+(A)/\|A\|_{1,1}$ for A an N x N matrix with simulated data from [9]. We randomly generate symmetric positive definite matrices and compute upper bounds on $\gamma_+(A)/\|A\|_{1,1}$ with different decompositions of A. The first step is generating

Gaussian distributed realizations in a matrix size N by N. Then by multiplying by its transpose, the result is symmetric positive semi-definite, denoted A. Let $\mathcal{A}_N$ denote a collection of 30 independent realizations of this random matrix.

We consider two factorizations of the matrix A: the LDL and the Eigen matrix decomposition. Specifically:

$$LDL: \quad A = \sum_{k=1}^{N} v_k v_k^*$$

with v_k vectors that have the top $k-1$ entries 0, and

$$Eigen: \quad A = \sum_{k=1}^{n} g_k g_k^*$$

where $\{g_1, ..., g_n\}$ are the eigenvectors, each scaled by the corresponding eigenvalue's square-root. For each decomposition denote:

$$J_{LDL}(A) = \sum_{k=1}^{N} \|v_k\|_1^2 \quad \text{and} \quad J_{Eigen}(A) = \sum_{k=1}^{N} \|g_k\|_1^2$$

Let F_{LDL} and F_{Eigen} denote the worst upper bounds over the N realization ensemble:

$$F_{LDL}(N) = \max_{A \in \mathcal{A}_N} \frac{J_{LDL}(A)}{\|A\|_{1,1}}$$

$$F_{Eigen}(N) = \max_{A \in \mathcal{A}_N} \frac{J_{Eigen}(A)}{\|A\|_{1,1}}$$

We plot these worst upper bounds after 30 realizations for various N in Fig. 3.1. In the same figure we plot the analytic approximations of these two curves using a square-root functions and a logarithmic function. The square-root function was scaled as $c\sqrt{N}$ to closely fit the Eigen decomposition bound, $F_{Eigen}(N)$. Numerically we obtained $c = 4/5$.

From these plots we notice a clearly strictly increasing trend. Furthermore, the LDL factrization produces a smaller (tighter) upper bound than the Eigen decomposition. On

the other hand, as shown in Theorem 2.9 in [9], any optimal decomposition may take N^2+1 vectors. By limiting the number of vector to N one should not expect to achieve the optimal bound $\gamma_+(A)$ with any decomposition.

Decomposition performance via sampling random matrices

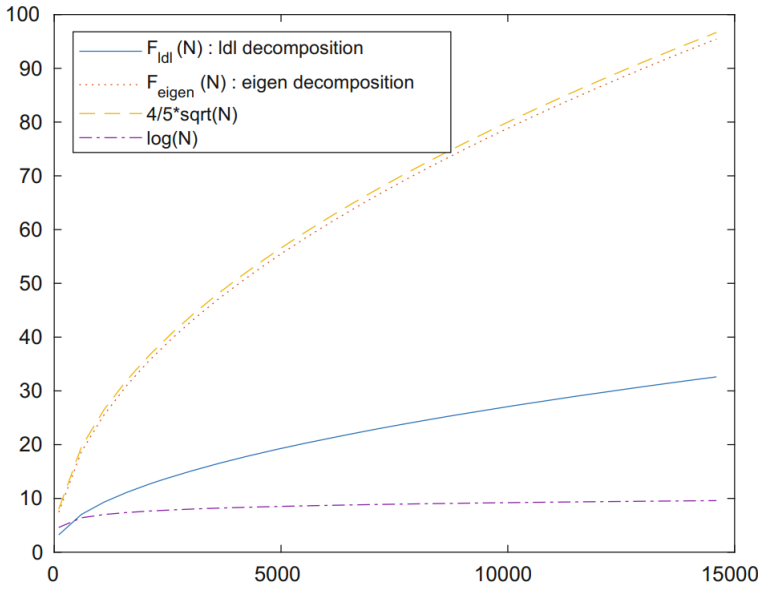


Figure 3.1: For each size N, 30 random matrices are sampled and decomposed in different ways. The worst upper bound of $\gamma_+(A)$ is plotted for various N. Reference curves are also plotted to indicate trend.

Numerical and statistical experiments show that LDL decompositions perform well with $\sum_{n\geq 1}\|g_n\|_1^2 \approx O(\log(N))$ where N is the size of the square matrix. Crucial to the analysis is the geometry of the space of matrices with the convexity and dimensionality necessary for the main theorems. We find geometric analysis for matrix decompositions, or more generally tensor decompositions, to be very useful and a promising direction for other matrix decomposition problems.

3.2 Open Problems

- Can the above theorems be generalized to rectangular matrices via the singular value decomposition, which corresponds to signals being mixed in complicated ways?

- Above we give the decomposition for diagonally dominant matrices, is there an approximate extension to *almost* diagonally dominant matrices?

Chapter 4

Optimal Transport Based Sparse Reconstruction

In the previous chapter we introduced operators and explored some of their properties. A mathematical problem written down in the late 1700's by French mathematician Gaspard Monge [68] posed: what is the most efficient operator to move a pile of material into a nearby hole?

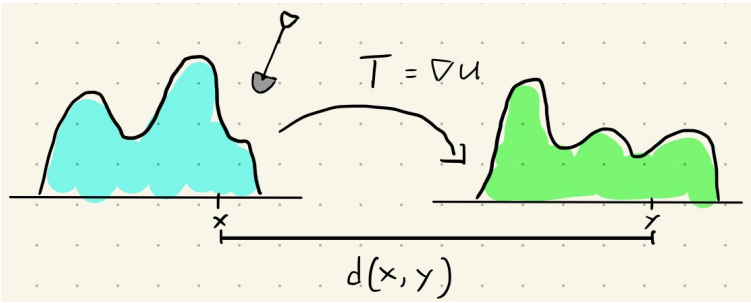


Figure 4.1: Image credit to Jakob Hultgren [53].

Mathematically, we describe a pile of material with a *probability distribution* and we describe a hole to fill with another *probability distribution*. Consider the blue mass on the

left and the green mass on the right to each be probability distributions in Fig. 4.1. Note we scale the mass by changing units so that the amount of mass is equal to 1. Call the distributions μ and ν on $\mathbb{R}$, then what is the most efficient operator T that moves the mass from μ to ν? By this we mean that for any ν-measurable set A, $\mu(T^{-1}(A)) = \nu(A)$ and $T^{-1}(A)$ is μ-measurable. This is the *pushforward measure*, written $T_*\mu := \mu \circ T^{-1}$. This condition forces $T : \mathbb{R} \to \mathbb{R}$ to map the mass from μ to exactly where ν has mass, which we called a hole to fill. We are close to a mathematical formulation of the problem now and we just need to define efficiency. From the intuition that moving mass 0 distance is the most efficient, we minimize $\int_{x\in\mathbb{R}} c(x, T(x))dx$ where a cost function c maps $\mathbb{R}\times\mathbb{R} \to \mathbb{R}$ and T must have $T_*\mu = \nu$. Finally, we have a well-specified formulation called the *Monge Formulation* and an optimal map T, if it exists, will tell us the most efficient way to move the material from a pile to a hole. More generally, for any space and measure pairs (X, μ) and (Y, ν), we can formulate the problem as

$$\inf_{T:X\to Y} \left\{ \int_X c(x, T(x))d\mu(x) : T_*\mu = \nu \right\}.$$

Of course solving this formulation is anything but simple. In some cases there is no solution. For example, the case when μ is a Dirac on $\mathbb{R}$ and ν is absolutely continuous on $\mathbb{R}$ has no map T where $T_*\mu = \nu$.

Centuries later, the problem came up in a time of allocating factory products for consumption [56]. Consider the simple case with 1 factory producing a stream of items and 2 consumers needing the items equally. A simple solution for the factory is to alternate back and forth between the 2 consumers. This solution does *not* satify the Monge Formulation because T must map the factory to exactly 1 consumer. That is the definition of a map. To

solve this problem, we need $T(f) = c_1$ just 50% of the time and $T(f) = c_2$ just 50% of the time where f is the factory source and c_1, c_2 are the consumers. A factory producing items does not need a fixed consumer, but can implement a *joint distribution* as above by allocating each item from sampling the distribution. Let γ be a joint distribution between space (X, μ) and space (Y, ν) then γ is a measure on $X \times Y$ where the marginals $\gamma(A \times Y) = \mu(A)$ and $\gamma(X \times B) = \nu(B)$ for any A μ-measurable and B ν-measurable. Now, we still want to minimize the cost but that should be scaled by the chance of the event. So we minimize the product $\int_{X \times Y} c(x, y) d\gamma(x, y)$, that is $\mathbb{E}_\gamma c(X, Y)$ where X, Y are random variables. This is known as the *Kantorovich Formulation* and the solution agrees with the Monge Formulation when there is a unique solution. Let $\Gamma(\mu, \nu)$ be the set of joint distributions, then we have

$$\inf_{\gamma \in \Gamma(\mu,\nu)} \left\{ \int_{X \times Y} c(x, y) d\gamma(x, y) \right\}. \tag{4.1}$$

When the Monge Formulation does not have a solution, the Kantorovich Formulation still has a solution, assuming cost c is lower semi-continuous and $\Gamma(\mu, \nu)$ is a *tight* collection of measures [108].

Not only does this solution tell one how to allocate items efficiently, it also tells one where to put the producers and the consumers in the first place. Take our above example with 1 factory and 2 consumers, let $f, c_1, c_2 \in \mathbb{R}$ be the locations of the factory and consumers. Then Equation (4.1) is $\frac{1}{2}|f - c_1|^2 + \frac{1}{2}|f - c_2|^2$ with $c(x, y) = |x - y|^2$. To minimize over f, set the derivative with respect to f to 0 by $(f - c_1) + (f - c_2) = 0$. So f equals the midpoint of c_1 and c_2, $f = \frac{c_1 + c_2}{2}$. The most optimal solution is of course $f = c_1 = c_2$, however constraints on consumer location may not allow this solution. In general, we have a *distance* between any distributions μ and ν on X which is known as the *Wasserstein* distance, or metric. For

any $p \in [1, \infty)$,

$$d_{W_p}(\mu, \nu) = \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \left\{ \int_{X \times X} d(u, w)^p \, d\gamma(u, w) \right\} \right)^{1/p} \tag{4.2}$$

defines a distance, or metric on distributions with finite p^{th} moments, written $\mathcal{P}_p(X)$, if X is a polish space and $d(\cdot, \cdot)$ is a metric on X [108].

So far we have seen how useful these formulations can be when solved, but we need a reliable method to minimize Equation (4.1). First we must introduce a dual formulation. Instead of minimizing Equation (4.1), maximize over functions $\phi \in L^1(X, \mu), \psi \in L^1(Y, \nu)$ with

$$\sup_{\phi(x)+\psi(y) \leq c(x,y)} \left\{ \int_X \phi(x) d\mu(x) + \int_Y \psi(y) d\nu(y) \right\}. \tag{4.3}$$

We can get a relation [108] with

$$\begin{aligned} \int_X \phi(x) d\mu(x) + \int_Y \psi(y) d\nu(y) &= \int_X \int_Y \phi(x) d\nu(y) d\mu(x) + \int_Y \int_X \psi(y) d\mu(x) d\nu(y) \\ &= \int_{X \times Y} \phi(x) + \psi(y) d\gamma(x, y) \leq \int_{X \times Y} c(x, y) d\gamma(x, y) \end{aligned}$$

by Fubini's Theorem [95], and since this holds for every $\gamma \in \Gamma(\mu, \nu)$ then

$$\sup_{\phi(x)+\psi(y) \leq c(x,y)} \int_X \phi(x) d\mu(x) + \int_Y \psi(y) d\nu(y) \leq \inf_{\gamma} \int_{X \times Y} c(x, y) d\gamma(x, y).$$

The next step is to prove equality which follows since if $\phi(x) + \psi(y) < c(x, y) - \epsilon$ on some measurable set A with $\epsilon > 0$ then $\phi'(x) := \phi(x) + \epsilon \mathbb{1}_A(x)$ would be a contradiction [108]. We shall use this dual formulation in the next algorithm to solve the optimization.

4.1 Discrete Optimal Transport & Sinkhorn Algorithm

One starts with a source distribution, $\mu \in \mathbb{P}(X)$, target distribution, $\nu \in \mathbb{P}(Y)$, and a cost $C : X \times Y \to \mathbb{R}$. In the discrete setting (where X and Y are finite sets) relevant to

scientific computing, distributions are represented as finite dimensional vectors. Given two vectors that represent distributions, $\mu \in \mathbb{R}^n, \nu \in \mathbb{R}^m$, together with a cost C represented as an $n \times m$ matrix (C_{ij}), the optimal transport plan between μ and ν is

$$\arg \min_{P \in \Pi(\mu,\nu)} \sum_{i=1}^{n} \sum_{j=1}^{m} C_{ij} P_{ij} \tag{4.4}$$

where $\Pi(\mu, \nu)$ is the set of transport plans from μ to ν

$$\Pi(\mu, \nu) = \{P \in \mathbb{R}^{n \times m} : \sum_{i=1}^{n} P_{ij} = \nu_j \ \forall j, \sum_{j=1}^{m} P_{ij} = \mu_i \ \forall i\}.$$

When $m = n$ and C defines a metric on $\{1, \ldots, n\}$ (i.e. $d(i, j) := C_{ij}$ is symmetric, non-degenerate, and satisfies the triangle inequality) then the minimum value

$$d_W(\mu, \nu) := \min_{P \in \Pi(\mu,\nu)} \sum_{i=1}^{n} \sum_{j=1}^{m} C_{ij} P_{ij} \tag{4.5}$$

defines a metric on $\mathbb{P}(\{1, \ldots, n\})$ [108].

For a probability mass function $p : \mathcal{J} \to \mathbb{R}$ (non-negative and sums to 1), we will use $H(p)$ to denote its entropy,

$$H(p) = -\sum_{\iota \in \mathcal{J}} p(\iota) \log(p(\iota)). \tag{4.6}$$

We will represent probability densities with vectors and matrices, so we consider the indices to be in the domain $\mathcal{J}$. In (4.6), we use the convention that $0 \cdot \log(0) = 0$. With this convention, H defines a continuous non-negative function on the probability simplex, differentiable in the interior of the probability simplex and with a derivative unbounded at the boundary.

The distance d_W in Equation (4.5) can be computed exactly using methods from linear programming. However, for large n the quickly growing computation times excludes this from many applications [25]. In applications involving large data sets (including machine

learning, vision, graphics and imaging) d_W is often replaced by its *entropic regularization*, which is much more feasible from a computational perspective [78]. Given a small constant $\epsilon > 0$, the ϵ-regularized distance between $\mu, \nu \in \mathbb{P}(\{1, \dots, n\})$ is

$$d_W^\epsilon(\mu, \nu) := \sum_{i,j} C_{ij} P^*_{ij} \tag{4.7}$$

where

$$P^* = \arg \min_{P \in \Pi(\mu,\nu)} \sum_{i,j} C_{ij} P_{ij} - \epsilon H(P). \tag{4.8}$$

The regularized objective function in (4.8) is strictly convex and proper, hence always admits a unique minimizer. While the minimizer in the true optimal transport distance d_W is usually very sparse, the entropy term in (4.8) pushes the minimizer away from the boundary of the unit simplex, producing a less sparse minimizer. As $\epsilon \to 0$, this minimizer converges to a minimizer of the original problem (4.5) (the minimizer with lowest entropy if there are more than one), and the ϵ-regularized distance d_W^ϵ converges to d_W [78].

A simple application of Lagrange multipliers show that the minimizer of (4.8) is the unique element in $\Pi(\mu, \nu)$ on the form

$$P_{ij} = \sum_{i,j} f_i e^{-c_i j/\epsilon} g_j \tag{4.9}$$

for some unknown positive multipliers $f = (f_1, \dots, f_n),\ g = (g_1, \dots g_n)$. Determining f and g from μ, ν and the matrix $(e^{-c_{ij}/\epsilon})$ is known as the matrix scaling problem, and a standard algorithm to find approximate solutions is the iterative proportional fitting procedure, also known as the Sinkhorn-Knopp Algorithm (or just Sinkhorn) [78]. The matrix (4.9) lies in $\Pi(\mu, \nu)$ if $\sum_i P_{ij} = \mu_i$ for all i and $\sum_j P_{ij} = \nu_j$ for all j. The Sinkhorn algorithm proceeds iteratively, alternating between updating f so that the first of these conditions is satisfied and updating g so that the second of these conditions is satisfied (see Algorithm 9).

The multipliers $f \in \mathbb{R}^n$ and $g \in \mathbb{R}^m$ above can be thought of as dual variables for the optimization problem (4.8). More precisely,

$$F = \epsilon \log(f),\ G = \epsilon \log(g)$$

are the Lagrange multipliers for (4.7) [25]. When considering the regularized distance between μ and ν as a function of μ (keeping ν fixed) this can, at least for positive μ and ν, be exploited to approximate its gradient (see for example [34, 59]). The Sinkhorn Algorithm, used to approximate $d_\mu^\epsilon(\mu, \nu)$ and its gradient with respect to μ is summarized in Algorithm 9. Note that the output F and G of Algorithm 9 needs to be projected onto the tangent space of the probability simplex to yield an approximation of the true gradients.

Algorithm 9: The Sinkhorn Algorithm for Regularized Optimal Transport Distances

Input:
$\mu, \nu \in \mathbb{R}^n$: positive probability vectors
$C \in \mathbb{R}^{n \times n}$: cost matrix
ϵ : positive regularization parameter
Output:
$d_W^\epsilon \in \mathbb{R}$: regularized distance between μ and ν
$F \in \mathbb{R}^n$: Gradient of $d_W^\epsilon(\mu, \nu)$ with respect to μ
$G \in \mathbb{R}^n$: Gradient of $d_W^\epsilon(\mu, \nu)$ with respect to ν
Begin:
$f = (1, \ldots, 1) \in \mathbb{R}^n$
$g = (1, \ldots, 1) \in \mathbb{R}^n$
while *f and g has not converged* **do**
  for $1 \leq i \leq n$ **do**
    $f_i = \mu_i / \left(\sum_j \exp(-C_{ij}/\epsilon) g_j \right)$
  for $1 \leq j \leq n$ **do**
    $g_j = \nu_j / \left(\sum_i \exp(-C_{ij}/\epsilon) f_i \right)$
$d_W^\epsilon = \sum_{i,j} f_i g_j \exp(-C_{ij}/\epsilon) C_{ij}$
$F = -\epsilon \log(f)$
$G = -\epsilon \log(g)$

We have introduced many aspects of optimal transport and discussed the metric and geometry that follows. We will apply this to provably reconstruct sparse signals in astronomical

measurements.

4.2 Star Cluster Detection Application

Star cluster detection from telescopic data can be performed with optimal transport [84]. Super resolution seeks to improve image resolution without further data collection. This is useful when important features or pixels are missing. Improving the measurement device, such as a camera or telescope, will improve the image resolution but only up to the diffraction limit governed by physical laws. Increasing image resolution beyond this point is possible but only with constraints that give a well-posed inverse problem. The most common constraint is that the image needs to be sparse or smooth, at least in some nontrivial basis.

For example, Gaussian noise is a common input that blurs important features. Removing additive Gaussian noise can be done, imperfectly, by solving an inverse problem that constrains total variation which enforces smoothness [73]. What about in non-smooth image domains? For magnetic resonance imaging problems [42], a solution using the Fourier basis is to extrapolate or interpolate in Fourier space. Then, transforming back to physical space gives a higher resolution image.

A more general solution is to minimize with respect to a regularizing term that maximizes sparsity. Compressed sensing methods often minimize an objective function involving the L^1 norm of the solution [16]. Minimizing L_0 maximizes sparsity but L^1 is usually used instead for its convex properties. Another regularizer that maximizes sparsity is entropy [23, 43]. Neural networks or deep learning has more recently been used for inverse problems, especially on images [74].

We propose two super resolution inverse problems that produce sparse solutions which are near to the measurement in Wasserstein distance. For a measurement ν and positive regularization parameters λ and λ', we define the *sparse approximation* of ν as a minimizer

$$\mu_* = \arg\min_{\mu \in \mathbb{P}(X)} d_W^\epsilon(\mu, \nu) + \lambda H(\mu). \tag{4.10}$$

and the *sparse retrieval* of ν as a minimizer

$$\mu_* = \arg\min_{\mu : d_W(\mu,\nu) < \lambda'} H(\mu). \tag{4.11}$$

Problem (4.10) and (4.11) are essentially dual, and for generic data ν there is a mapping $\lambda \mapsto \lambda'(\lambda)$ such that μ is a solution to (4.10) if and only if μ is a solution to (4.11). We will approach (4.11) from a theoretical perspective below but use (4.10) in our application since it fits well into a gradient descent method.

At least one minimizer exists by compactness of the finite dimensional probability simplex. The entropy term in (4.10) favors sparse solutions. Naturally, there is a trade-off between sparsity of the solution and proximity to the measurement. How these two objectives are prioritized is governed by λ. For $\lambda = 0$, no priority is given to the goal of sparsity and $\mu_* = \nu$. As λ increases, μ_* turns into an increasingly sparse approximation of ν and when $\lambda \to \infty$, $\|\mu_*\|_0 \to 1$.

This inverse problem is useful whenever there is a natural distance, or cost function, on the index set of ν. If, for example, ν is given in Fourier space and each entry μ_i corresponds to a frequency σ_i, then two natural choices for the cost C_{ij} are $C_{ij} = |\sigma_i - \sigma_j|$ and $C_{ij} = |\log(\sigma_i/\sigma_j)|$. In the application we describe below, each entry in ν describes the intensity of a pixel in a 32×32 image and C_{ij} is chosen as the L^2-distance between the i^{th} pixel and j^{th} pixel.

Remark. *This method can be contrasted to maximum entropy methods in statistical physics, where the probability distribution with highest entropy (under constraints dictated by observations) is chosen as the best representative of the current state of knowledge about a system. In our context, we work with the crucial assumption of sparsity, which motivates minimizing the entropy instead of maximizing it.*

We will let probability vector $\nu \in P(\{1, \ldots, n\})$ be a sparse signal (i.e. $\|\nu\|_0 < n$ is small), and use $\bar{\nu}$ to denote this signal with noise and distortion. In our application, we are interested in determining the support of ν (i.e. the indices of all non-zero entries in ν) from $\bar{\nu}$. For two probability vectors μ and ν we will say that μ identifies the structure of ν if they have the same support, i.e. if $\mu_i > 0$ if and only if $\nu_i > 0$ for all i. Our main theorem (Theorem 11 below) shows that the minimizer of (4.10) identifies the structure of ν under the assumptions that ν is sparse and the noisy signal $\bar{\nu}$ is close to ν in optimal transport distance. As is indicated by Theorem 10 below, the latter assumption is natural when dealing with Gaussian noise since the optimal transport distance, unlike total variation and L^p distances, take the geometry of the space into account.

Let the probability distribution $\nu := \frac{1}{k}\sum_{i=1}^{k} \delta_{p_i}$ be a sparse signal in $\mathbb{P}(\mathbb{R}^d)$ where δ is the Dirac delta. Assume the noisy signal $\tilde{\nu}$ is produced in the following way: For each p_i in the sum above, we sample n points $x_i^1, \ldots, x_i^n$ in $\mathbb{R}^d$ according to a normal distribution centered at p_i with independent components of variance σ^2. Let $N = kn$ be the number of points sampled and $\tilde{\nu} = \frac{1}{N}\sum_{i=1}^{n}\sum_{j=1}^{m} \delta_{x_i^j}$ be the noisy signal.

Theorem 10. *[84] Given a sparse signal $\nu \in \mathbb{P}(\mathbb{R}^d)$ giving rise to a noisy signal $\tilde{\nu}$ as described above, the optimal transport distance between ν and $\tilde{\nu}$ is bounded by $\frac{\sigma^2}{N} X_N$ where*

X_N is a random variable with distribution χ^2_{dN}. In particular, the expected value and variance of $\frac{\sigma^2}{N}X_N$ are $d\sigma^2$ and $2d\sigma^4/N$, respectively.

Proof. The optimal transport cost is bounded from above by the cost of the transport plan sending each x_i^j to p_i. The cost of this plan is $\frac{1}{N}\sum |x_i^j - p_i|^2$. By assumption, each term in this sum is the squared sum of d normal distributed random variables with mean 0 and variance σ^2. □

Theorem 11. *[84] Assume ν is a sparse signal and $\bar{\nu}$ is a noisy signal such that $d_w(\nu, \bar{\nu}) < \delta$. Then the solution of*

$$\mu = \arg \min_{\mu: d_W(\bar{\nu}, \mu) \leq \delta} H(\mu) \tag{4.12}$$

will identify the structure of ν, i.e. have the same support as ν, if $||\nu||_0 \leq ||\mu||_0$ for all μ such that $d_W(\mu, \bar{\nu}) < 2\delta$, with equality only if μ and ν has the same support.

Remark. *The conditions in Theorem 11 can be summarized as a low enough noise level δ and enough sparsity of the true signal ν (making it a local minimizer of the L^0-norm). It is interesting to note that these conditions are essentially necessary: if the inequality in Theorem 11 is violated by some μ closer than δ to $\bar{\nu}$, then the solution of* (4.12) *does not identify the structure of ν.*

Remark. *Noise is high entropy, hence it is expected that the noise can be removed by minimizing the entropy. However, if the signal-to-noise ratio is too low, this reconstruction is underdetermined.*

Proof of Theorem 11. By the triangle inequality, the feasible set in (4.12) is contained in the ball centered at $\bar{\nu}$ of radius 2δ. As the feasible set in (4.12) contains ν, this means any solution of (4.12) has to be ν or have the same support as ν. □

Theorem 12. *[84] Fix a positive probability vector $\nu \in \mathbb{R}^d_{>0}$ such that all elements of ν are distinct. Then the sparse recovery is continuous to perturbations around ν for small λ, i.e. for every $\epsilon' > 0$ there exists $\delta > 0$, such that if $d_W(\nu, \nu') < \delta$,*

$\mu_* = \arg\min_{\mu \in \mathbb{P}(X): d_W(\mu,\nu) < \lambda} H(\mu)$, *and*

$\mu'_* = \arg\min_{\mu \in \mathbb{P}(X): d_W(\mu,\nu') < \lambda} H(\mu)$ *then* $\|\mu_* - \mu'_*\| < \epsilon'$.

Proof. The assumption on ν guarantees that minimizers are unique for small λ. Continuity of the minimizer then follows from smoothness of H. □

We solve (4.10) using a gradient descent method with variable step size. More precisely, letting $J(\mu) := d^{\epsilon}_W(\mu, \nu) + \lambda H(\mu)$ be the objective we set the step size to $\alpha_* := \sup\{\alpha > 0 : J(\mu - \alpha \nabla J|_\mu) < J(\mu)\}$. As mentioned in above, the entropy is not differentiable on the boundary of the probability simplex. An effect of this is that the output F in Algorithm 9 is infinity in all indices where μ is zero. We circumvent this problem by defining the i^{th} entry in the gradient of J to be 0 whenever $\mu_i = 0$. Geometrically, this means that whenever the algorithm reaches a sub-simplex of the probability simplex, it ignores the component of the gradient orthogonal to this sub-simplex, thus remaining in this sub-simplex for the rest of the algorithm. Gradient descent will converge to a local minimum on the compact probability simplex since the objective is smooth when restricted to the local simplex face. Algorithm 10 contains the pseudocode and also predicts the star cluster classification.

Algorithm 10: Optimal Transport Star Cluster Prediction

Input:

$X \in \mathbb{R}^{N\times m\times m}$: N images size $m \times m$

$\lambda \in \mathbb{R}$: positive noise level

$0 < \epsilon < 1$: optimal transportation regularization

$C \in \mathbb{R}^{m^2\times m^2}$: cost matrix

$J_{\lambda,\epsilon}(x, v) := d^{\epsilon}_W(x, v) + \lambda H(v)$

Output:

$K \in \mathbb{R}^N$: star cluster classification

Begin:

for $i = 1, 2, ..., N$ **do**

  $v = X_i$

  while *v has not converged* **do**

    $w = \nabla d^{\epsilon}_W(X_i, \cdot)|_v + \lambda \nabla H|_v$

    $w = w - \langle w, \frac{1}{m}\mathbb{1}\rangle \cdot \frac{1}{m}\mathbb{1}$

    $\alpha = \sup\{\alpha \in \mathbb{R} : J_{\lambda,\epsilon}(v) > J_{\lambda,\epsilon}(v - \alpha w)\}$

    $\alpha = \min\{0.01, \alpha\}$

    $v = v - \alpha w$

    $v = diag(\mathbb{1}_{v>0})\ v$

    $v = v/\|v\|_1$

  $V_i = v$

  $\delta = \max V_i$

  if $rank(H_0(V_i^{-1}([0.75\delta,\ \delta]))) == 1$ **then**

    $K_i = 1$

  else

    $K_i = 0$

4.2.1 Simulation

We first show this method's results on a low dimensional example. For example, let the measurement be

$$\nu = (0.2, 0.15, 0, 0, 0, 0.1, 0.15, 0.2, 0.15, 0.1)^T.$$

With sparsity parameter $\lambda = 10$ the method produces the sparse approximation

$$\mu_* = (0.35, 0, 0, 0, 0, 0, 0, 0.65, 0, 0).$$

This reflects the fact that ν has two peaks, one peak centered at position 1 and one peak centered at position 8, and that 35% of the mass of ν is situated close to position 1 and

65% of the mass of ν is situated close to position 8. Fig. 4.2 plots ν, the gradient descent steps, and the final result. With sparsity parameter $\lambda = 100$, the method produce the sparse approximation $(0, 0, 0, 0, 0, 0, 0, 1, 0, 0)$, reflecting the fact that most of the mass of ν is part of a peak centered at position 8. Fig. 4.3 plots ν, the gradient descent steps, and the final result.

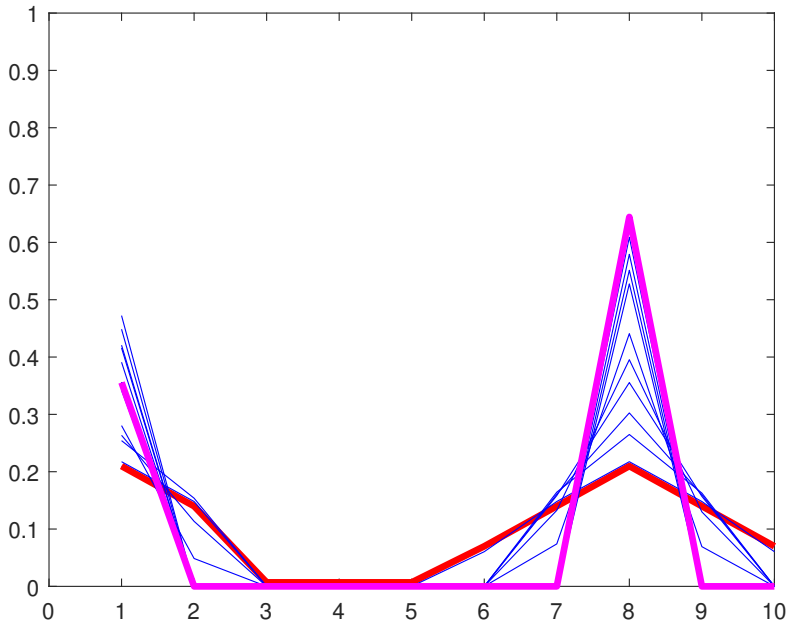


Figure 4.2: Plot of super resolution O.T. method Algorithm 10. Red line is initial distribution. Blue lines are steps along gradient of Equation (4.10). Pink line is final, converged distribution. $\lambda = 10$. epsilon = 0.1. Max Sinkhorn iterations = 5000. Gradient step size = 0.01. Gradient steps=50.

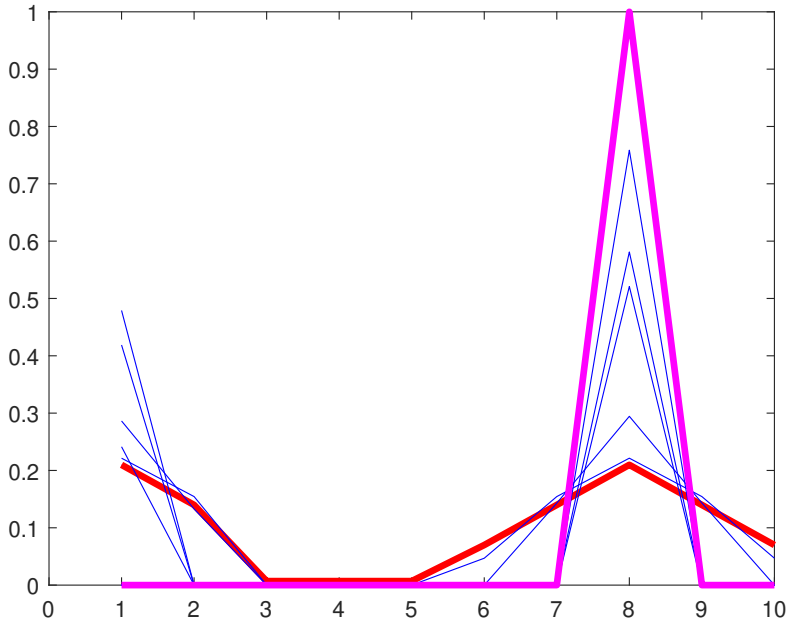


Figure 4.3: Plot of super resolution O.T. method Algorithm 10. Red line is initial distribution. Blue lines are steps along gradient of Equation (4.10). Pink line is final, converged distribution. $\lambda = 100$. epsilon $= 0.1$. Max Sinkhorn iterations $= 5000$. Gradient step size $= 0.01$. Gradient steps=50.

4.2.2 Astronomy Data

Next, we will analyze the real data. The formation and evolution of star clusters provide insight into the processes governing the birth of stars as well as the dynamical evolution of galaxies [81]. In order to save human hours and get reproducible results, it is of interest to algorithmically detecting star clusters in images of sky patches. Many methods have been proposed to algortihmically detect star clusters, including CLEAN [54], Multiscale CLEAN [24], IUWT-based CS [61], decision trees [40] and optimal sheaves [93]. The state of the art method trains a convolutional neural network (CNN) to classify each sky patch or region

in an image as containing a star cluster or not [81]. These neural networks are notoriously computationally expensive, sensitive to noise, and inflexible to appending or removing data variables.

We propose using the Wasserstein inverse problem (4.10) to detect star cluster locations. Our dataset consists of measurements from the Hubble Space Telescope in the survey Treasury Project LEGUS (Legacy ExtraGalactic Ultraviolet Survey) [14]. This data set consists of 32×32 pixel images of star patches. Each image comes in 5 frequency bands (NUV, U, B, V, and I) [14]. We encode each of these in a probability vector ν where each entry correspond to the intensity of a pixel, normalized to sum to 1. Algorithm 10 produces a sparse approximation each image which is classified as a star cluster if it contains just one 'peak'. Specifically, the sparse image is made binary (1 and 0) by thresholding at 75% of the max of the image. Then the number of 'peaks' is the number of connected components in the binary image. This is the rank of the 0^{th} homology of the binary image, denoted $rank(H_0(V_i^{-1}([0.75\delta,\ \delta])))$ in Algorithm 10. We perform this calculation for each of the 5 frequency bands that were measured. Then these 5 predictions vote to produce the final prediction for the image in question.

Algorithm 10 can be compared to the method of producing a binary image directly from the source image, without first producing a sparse approximation, and counting the number of connected components in this. The accuracy rate of this naive approach is 46% with respect to the CNN. Algorithm 10 increases in accuracy to 74% with respect to the CNN. The CNN accuracy rate is 86% with respect to experts, but even experts agree with each other only around 70%-75% [2, 40, 111]. Given that experts are the baseline, it is impossible, without overfitting, for a computational model to do better than that. Therefore our method

provides a very high performance given that no neural network training, which often takes weeks of compute time, is required. Additionally, the O.T. method is less sensitive to noise than a CNN, see [121], which we describe and bound in Theorem 10, Theorem 11, and Theorem 12. Finally, with our method, variables can be simply added and removed where Equation (4.10) is quickly recalculated.

We give the confusion matrix in Table 4.1 from our calculations. We test on 128 random samples. The maximum Sinkhorn iterations is 500. The cost C_{ij} is chosen as the L^2-distance between the i^{th} pixel and j^{th} pixel. The H_0 threshold is 0.75. The initial gradient descent step size is 0.001. Wasserstein parameter $\epsilon = 0.001$. Sparsity parameter $\lambda = 1$. When specifying the accuracy rates in the previous paragraph we use the classification results of the CNN in [81] as the definition of the correct classification.

Table 4.1: Confusion matrix of O.T. metod Algorithm 10 on LEGUS data compared to StarcNet [81]. Column gives StarcNet classification and row gives Algorithm 10 classification.

	StarcNet Cluster	StarcNet Not Cluster
O.T. Cluster	25% (32)	13.3% (17)
O.T. Not Cluster	12.5% (16)	49.2% (63)

Optimal transportation is more efficient, robust, and flexible than CNNs. We proved that optimal transportation will reconstruct sparse sources and is robust to noise. This is relevant for correcting distortions and noise in imaging which we showed for star cluster detection. Another benefit of a predictive model for star clusters is that it can produce a *policy* that informs where future surveys should look for star clusters [33, 82].

4.2.3 Open Problems

- How can global optimization be performed and guaranteed?

- Can local convergence be done with higher order (faster)?

Chapter 5

Fourier Space Reconstruction with MRI Motion Correction Applications

Periodicity is a key concept in pattern recognition. A periodic signal, that is a signal that repeats on some interval, may be a pattern that ought to be recognized. If a signal, $f : \mathbb{R} \to \mathbb{C}$, is periodic, then the signal can be viewed as a function or signal on a circle, $\phi : S^1 \to \mathbb{C}$. We say the f 'factors' through S^1 via $f(x) = \phi(\cos(\alpha x), \sin(\alpha x))$. We can write this with a *commutative diagram* [6] as follows

$$\begin{array}{ccc} \mathbb{R} & \xrightarrow{f} & \mathbb{C} \\ {\scriptstyle (\cos(\alpha x),\sin(\alpha x))}\downarrow & \nearrow_{\phi} & \\ S^1 & & \end{array} .$$

Then periodicity just depends on the *topology* of the domain, in this sense. And if the signal f is continuous then the image $\phi(S^1)$ is a compact set [70]. If f is an embedding then $\phi(S^1)$ has the topology of S^1 [58]. And then image $f(\mathbb{R})$ has the topology of S^1 too. Sampling and then estimating the topology of $f(\mathbb{R})$ with persistent homology can indicate periodicity, see

Chapter 6.

Once the periodicity of a continuous signal, f, is confirmed, it is possible to decompose the signal into the exponential basis [11]. With period P, let $f(x) = \sum_{k\in\mathbb{Z}} \alpha_k e^{2\pi i \frac{k}{P} x}$. Since the coefficients α_k depend on f, we'll write them as $\widehat{f}(k) = \alpha_k$ and call this 'f in k-space'. By orthogonality, $\widehat{f}(k) = \langle f(x), e^{2\pi i \frac{k}{P} x} \rangle = \frac{1}{P} \int_0^P f(x) e^{-2\pi i \frac{k}{P} x} dx$. We define the analogous transform for function $f \in L^1(\mathbb{R}, \mathbb{C})$ as $\widehat{f}(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx$. We call the former the 'discrete Fourier transform' (DFT) and the latter the continuous Fourier transform [11]. We will often write this transform as a map $\mathcal{F} : L^1 \to L^1$ where $\mathcal{F}(f)(k) = \widehat{f}(k)$. Reconstructing signal f from samples of $\widehat{f}$ is often useful in applications. We will discuss a Magnetic Resonance Imaging (MRI) application where we reconstruct f under challenging circumstances with heavily undersampled measurements of $\widehat{f}$.

MRI is one of the most common and useful medical devices. An MRI machine can detect or image 3D bodies within the machine. Unlike a traditional image that is 2D, an MRI image is 3D and a 2D image at each depth can be rendered. MRI machines accomplish this with sensitive electromagnets. A powerful magnet is designed in an outer annulus. Within it are the Gradient Coils in another annulus and within that are the Radio Frequency (RF) Coils in another annulus. The center of the annuli is the empty Bore. The Bore is loaded with some subject for imaging. A diagram of an MRI machine is shown in Fig. 5.1 [49].

The outer magnet aligns the subject's nuclei spin that have nuclear magnetic moments [49]. A current pulse in the Radio Frequency Coils produces a magnetic field in the orthogonal direction which reorients the spin temporarily. The Radio Frequency Coils turn off and the Radio Frequency Coils detect the change in subject spin, via current, to the original direction. The speed of the change indicates the type of molecule via its magnetic moment.

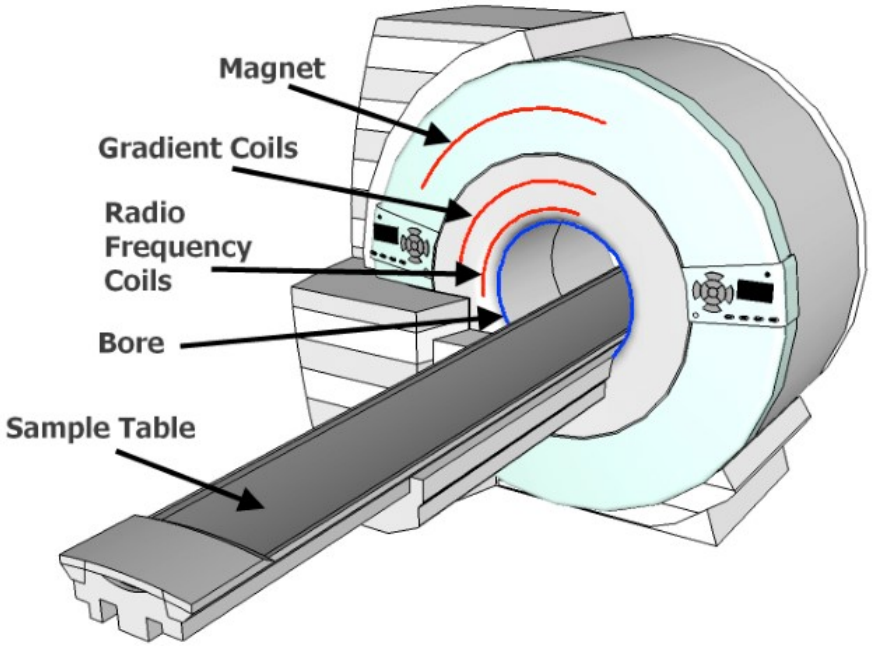
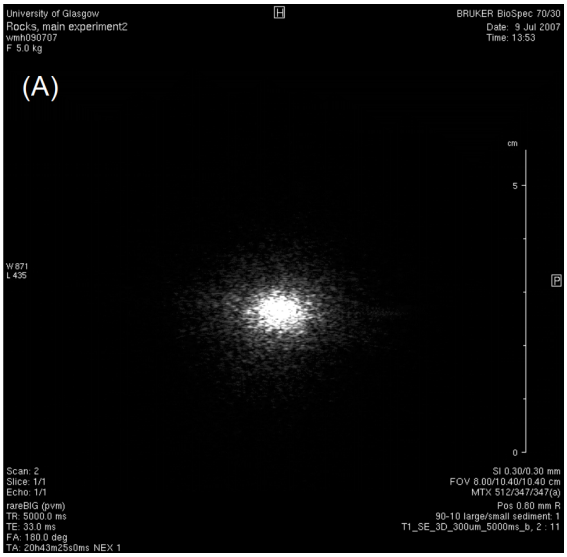


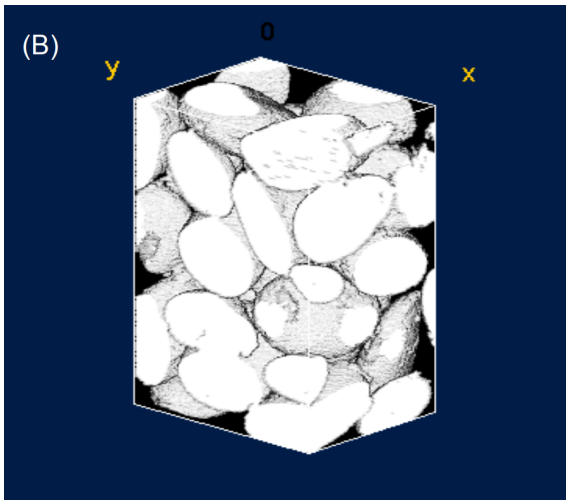
Figure 5.1: Magnetic Resonance Imaging (MRI) Machine. Image from [49].

The Gradient Coils also play an important role. The subject's nuclei spin oscillate at the Larmor frequency about the static magnetic field. The Gradient Coils are used to vary the static magnetic field and thus vary the Larmor frequencies in space. The RF Coils will sample a specific frequency and thus sample a specific 3D region in the Bore. Another key point is that each RF Coil can sense nearby nuclei stronger than distance nuclei. This is the r^2 law and so we model each measurement as a product of the nuclei signal and the sensitivity of the RF Coil (also called the profile of the RF Coil). For example, in Fig. 5.2a we show the magnitudes of different frequencies sampled by an MRI machine [49]. This is a 2D slice, but putting the 2D slices together and calculating the 3D inverse Fourier transform give us the true image, visualized in Fig. 5.2b [49].

A common technique to speed up MRI scans is to skip collecting certain data points, which is important for various reasons. However, with missing Fourier coefficients, the in-



(a) Raw Data, 2D slice (Modulus of FFT Coefficients).



(b) 3D Inverse Fourier Transform. Plot of sediment in water.

Figure 5.2: MRI data locates Hydrogen in liquid water. Image from [49].

verse Fourier transform cannot be computed and the subject image is not produced. A successful technique to approximate the missing data called Generalized Autocalibrating Partially Parallel Acquisition (GRAPPA) was introduced in 2002 [42]. GRAPPA approximates missing data lines (constant k_y value) in Fourier space, that are away from the center or the low frequencies. We plot example missing data lines in red in Fig. 5.3.

The missing data will be approximated by a linear combination of nearby samples (non-missing data). Now, the fully sampled lines near the center or constant exponential are called the Auto-Calibration Signal (ACS). The weights for the above linear combination are calculated with the ACS samples. Specifically, GRAPPA optimizes a set of weights to equate ACS samples with their neighbors. We visualize this in Fig. 5.4.

Next, we will write down the specific optimization problem mathematically. Let the

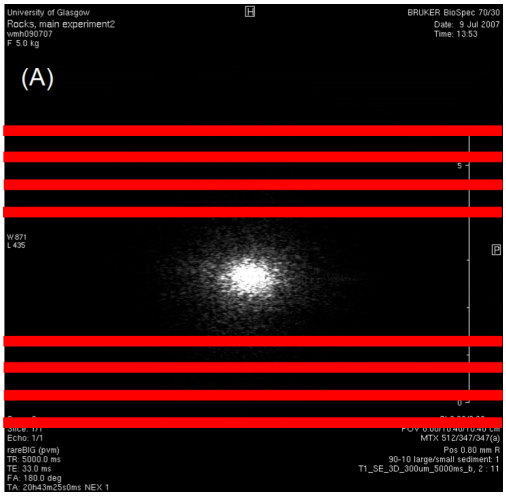


Figure 5.3: MRI data, 2D slice, with missing data in red. Image from [49].

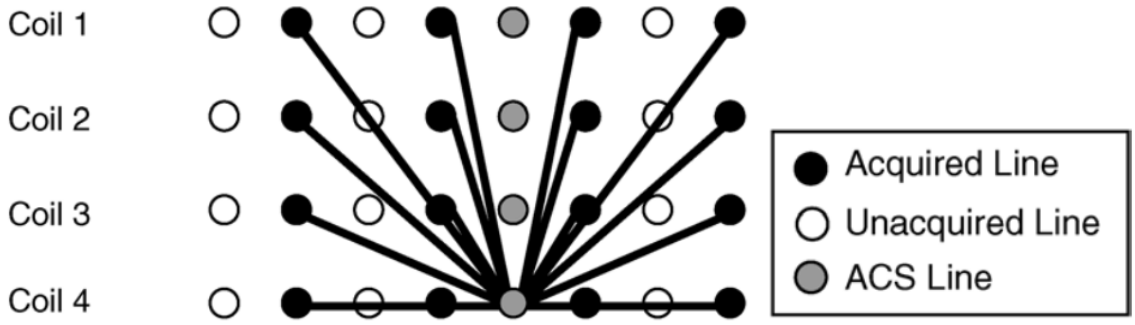


Figure 5.4: GRAPPA. Image from [42].

number of RF Coils be k. Let the stencil width (neighborhood size) be s_w and the interval be s_i. Let resolution of the image be n. Let auto-calibration signal, $ACS \in \mathbb{C}^{n_0 \times n}$. Let the weights be $\omega \in \mathbb{R}^{k \times n \times s_w \times k}$ and let $\Phi = \{(u,v) \in ACS : stencil(u,v) \in Acquired\}$. Then GRAPPA will optimize

$$\min_{\omega} \sum_{(u,v)\in\Phi} \sum_{j=1}^{k} \left| \widehat{P_j f}(u,v) - \sum_{l=1}^{k} \sum_{b=-s_w/2}^{s_w/2-1} \omega_{j,u,b,l} \widehat{P_l f}(u, v + s_i/2 + b s_i) \right|^2$$

To solve this system of linear equations, apply the pseudoinverse.

GRAPPA is a hugely successful method that is used frequently. However, subject motion during a scan, especially with living subjects, breaks the assumptions of GRAPPA.

Occasional subject motion is inevitable during MRI scans. Motion artifacts can be partially corrected using post-processing or prospective correction methods [65]. Here, we consider motion-induced image reconstruction errors for parallel imaging (PI) [102]. For time series imaging, calibration, or ACS, data are often acquired in the beginning of the entire imaging process to save time and to keep the image contrast the same for all image frames. When motion occurs after the calibration data acquisition, the pre-motion calibration data no longer match the spatial position and the corresponding coil sensitivity encoding of the subsampled data. This mismatch subsequently causes image reconstruction errors using the reconstruction kernels estimated from the calibration data. This problem can be solved by re-acquiring calibration data and re-estimating the PI reconstruction coefficients, but this solution is commonly impractical and requires additional scan time. The purpose of this study is to propose a new method to solve this problem without reacquiring calibration data. As a proof-of-concept study, we focused on PI reconstruction based on Cartesian sub-sampling and GRAPPA [42, 110] but the solution can be extended into non-Cartesian sampling and non-GRAPPA PI reconstruction schemes. Our method [89], which we call Motion-corrected GRAPPA, or MGRAPPA, approximately accounts for and correction motion caused reconstruction errors. MGRAPPA uses prospective motion correction (PMC) [65] measurements to approximate changes in coil sensitivity maps.

Fig. 5.5 shows a flowchart of MGRAPPA in a typical PI scan. For each RF coil, a fully sampled calibration data set (centered in k-space) and an under-sampled data set are acquired. GRAPPA recovers missing k-space data by convolving the sub-sampled data with kernels calculated from calibration data. For MGRAPPA, the acquisitions remain unchanged, but the GRAPPA kernels are updated by reconstructing low-resolution coil sen-

sitivity maps from the calibration data, transforming sensitivity maps inversely to subject motion, and recalculating calibration data and finally GRAPPA kernels.

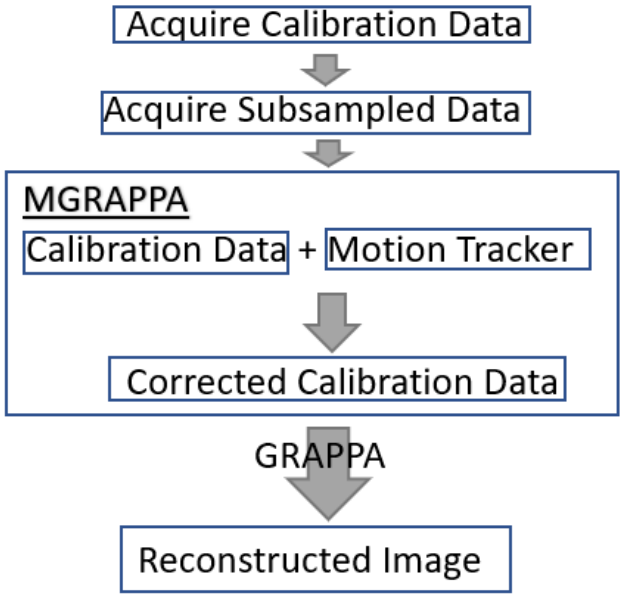


Figure 5.5: Diagram showing flow of stages of scan and image reconstruction.

We will describe the method mathematically. Denote the Fourier transform operator by $\mathcal{F}$. For any signal $f : \mathbb{R}^2 \to \mathbb{R}$, its Fourier transform is denoted by $\hat{f}$ or $\mathcal{F}(f)$ and is given explicitly by:

$$\hat{f}(u,v) = \mathcal{F}(f)(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-2\pi i(ux+vy)} f(x,y)dxdy$$

Each coil $i \in [1,n]$ returns a data matrix obtained by sampling the Fourier transform of $P_i^{0,0,0}f$, where $P_i^{(0,0,0)}$ is the i^{th} coil profile and f is the signal of interest. Define $L(g) := \mathcal{F}^{-1}(\hat{g} \cdot \mathbb{1}_{[-\alpha,\alpha]^2})$ the *lowpass* operator associated to the frequency band $[-\alpha,\alpha] \times [-\alpha,\alpha]$. We measure $L(P_i^{0,0,0}f)$ for each i. First we approximate the low frequencies of the signal, f, (ex: brain) by

$$L(f)(x,y) \approx \tilde{f}^l(x,y) := \sqrt{\sum_{i=1}^{n} |L(P_i^{0,0,0}f)(x,y)|^2}.$$

The first Theorem provides us with an upper bound on the approximation error.

Theorem 13. *[89] For some $\alpha > 2\beta > 0$ assume that: (1) supp$(\hat{P}_i) \subset [-\beta, \beta]$; (2) $P_i(x, y) \geq 0$, and (3) the coil profiles define a squared partition of unity, i.e., $\sum_i P_i(x, y)^2 = 1$ for every x, y. Let $L(g) := \mathcal{F}^{-1}(\hat{g} \cdot \mathbb{1}_{[-\alpha,\alpha]^2})$ denote the lowpass operator associated to the frequency band $[-\alpha, \alpha] \times [-\alpha, \alpha]$. Let $f : \mathbb{R}^2 \to \mathbb{R}$ be a signal such that $f(x, y), L(f)(x, y), L(P_i f)(x, y)$, and $P_i(x, y)$ are greater than $\gamma > 0$ for all x, y. Let $S = (-\alpha + 2\beta, \alpha - 2\beta)^2$, $M = [-\alpha - 2\beta, \alpha + 2\beta]^2 \backslash S$, and $H := \mathbb{R}^2 \backslash [-\alpha - 2\beta, \alpha + 2\beta]^2 = (\mathbb{R}^2 \backslash M) \backslash S$.*

Then for $\sup_{u,v} |\hat{f}(u, v) \cdot \mathbb{1}_M(u, v)|$ small enough,

$$\left| L(f)(x, y) - \sqrt{\sum_{i=1}^{n} |L(P_i f)(x, y)|^2} \right| \leq O\left(\sup_{u,v} |\hat{f}(u, v) \cdot \mathbb{1}_M(u, v)| \right).$$

Remark. *The approximation $\tilde{f}^l(x, y)$ is reasonable because the coil profiles, P, (superscripts are rotation, θ, and translation, x, y, of coil profile and subscript is coil profile index) are approximately a partition of unity and lowpass, denoted L (L clears high frequency coefficients of the Fourier coefficients), is linear.*

First, we will state and prove a lemma.

Lemma 14. *[89] Assume the assumption of Theorem 13.*

$$L(P_i f)^2 - (P_i f_S)^2 = (\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 + (P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))$$

and

$$|L(P_i f)^2 - (P_i f_S)^2| \leq O(\sup_{(u,v)} |\hat{f}(u, v) \cdot \mathbb{1}_M(u, v)|).$$

Proof of Lemma 14. Let

$$\hat{f}_S = \hat{f} \cdot \mathbb{1}_S, \ \hat{f}_M = \hat{f} \cdot \mathbb{1}_M, \ \hat{f}_H = \hat{f} \cdot \mathbb{1}_H$$

So $\hat{f} = \hat{f}_S + \hat{f}_M + \hat{f}_H$.

$$\mathcal{F}(P_i f) = \hat{P}_i * \hat{f} = \hat{P}_i * \hat{f}_S + \hat{P}_i * \hat{f}_M + \hat{P}_i * \hat{f}_H$$

$$L(P_i f) = \mathcal{F}^{-1}((\hat{P}_i * \hat{f}_S) \cdot \mathbb{1}_{[-\alpha,\alpha]^2} + (\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2} + (\hat{P}_i * \hat{f}_H) \cdot \mathbb{1}_{[-\alpha,\alpha]^2})$$

$$= \mathcal{F}^{-1}(\hat{P}_i * \hat{f}_S + (\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2})$$

$$= P_i f_S + \mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2})$$

Square both sides

$$L(P_i f)^2 = (P_i f_S)^2 + (\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 + 2(P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))$$

So

$$|L(P_i f)^2 - (P_i f_S)^2| \leq O(\sup |\hat{f}_M|).$$

□

Proof of Theorem 13. Recall $S = (-\alpha + 2\beta, \alpha - 2\beta)^2$, $M = [-\alpha - 2\beta, \alpha + 2\beta]^2 \backslash S$, and $H = \mathbb{R}^2 \backslash M \backslash S$. Let

$$\hat{f}_S = \hat{f} \cdot \mathbb{1}_S, \ \hat{f}_M = \hat{f} \cdot \mathbb{1}_M, \ \hat{f}_H = \hat{f} \cdot \mathbb{1}_H$$

Notice $\hat{f} = \hat{f}_S + \hat{f}_M + \hat{f}_H$. From Lemma 14

$$\begin{aligned} L(P_i f)^2 = (P_i f_S)^2 + (\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 \\ +2(P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2})) \end{aligned} \tag{5.1}$$

Then

$$\begin{aligned}
&\sum_{i=1}^{n} L(P_i f)^2 = \sum_{i=1}^{n} (P_i f_S)^2 \\
&+ \sum_{i=1}^{n} [(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 + 2(P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))] \\
&= \sum_{i=1}^{n} P_i^2 f_S^2 + \sum_{i=1}^{n} [(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 + 2(P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))]
\end{aligned} \tag{5.2}$$

from the square partition of unity

$$\begin{aligned}
&= f_S^2 + \sum_{i=1}^{n} [(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 + 2(P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))] \\
&= (\mathcal{F}^{-1}(\hat{f} \cdot \mathbb{1}_{[-\alpha,\alpha]^2}) - \mathcal{F}^{-1}(\hat{f} \cdot \mathbb{1}_{[-\alpha,\alpha]^2 \setminus S}))^2 \\
&+ \sum_{i=1}^{n} [(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 + 2(P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))]
\end{aligned} \tag{5.3}$$

$$\begin{aligned}
&= (L(f) - \mathcal{F}^{-1}(\hat{f} \cdot \mathbb{1}_{[-\alpha,\alpha]^2 \setminus S}))^2 \\
&+ \sum_{i=1}^{n} [(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 + 2(P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))] \\
&= L(f)^2 + (\mathcal{F}^{-1}(\hat{f} \cdot \mathbb{1}_{[-\alpha,\alpha]^2 \setminus S}))^2 - L(f) \cdot (\mathcal{F}^{-1}(\hat{f} \cdot \mathbb{1}_{[-\alpha,\alpha]^2 \setminus S})) \\
&+ \sum_{i=1}^{n} [(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 + 2(P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))]
\end{aligned} \tag{5.4}$$

Let

$$\begin{aligned}
T_1(f) &:= (\mathcal{F}^{-1}(\hat{f} \cdot \mathbb{1}_{[-\alpha,\alpha]^2 \setminus S}))^2 - L(f) \cdot (\mathcal{F}^{-1}(\hat{f} \cdot \mathbb{1}_{[-\alpha,\alpha]^2 \setminus S})) \\
&+ \sum_{i=1}^{n} [(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 + 2(P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))]
\end{aligned} \tag{5.5}$$

Square root to get

$$\begin{aligned}
\sqrt{\sum_{i=1}^{n} L(P_i f)^2} &= \sqrt{L(f)^2 + T_1(f)} \\
&= L(f) + T_1(f) \cdot \xi^{-1/2}/2
\end{aligned}$$

for some $|\xi - L(f)^2| < |T_1(f)|$ and $0 < \xi$. For f_M small enough, $L(f)^2 - T_1(f) > L(f)^2/2$.

Then

$$\left| L(f) - \sqrt{\sum_{i=1}^{n} |L(P_i f(x,y))|^2} \right| \leq |T_1(f) \cdot \xi^{-1/2}/2|$$
$$\leq |T_1(f) \cdot (L(f)^2 - T_1(f))^{-1/2}/2|$$
$$\leq \frac{|T_1(f)|}{2\sqrt{L(f)^2/2}}$$
$$\leq O\left(\sup |\hat{f}_M|\right).$$

□

Next we utilize extrapolation. Let $E_\delta(P)$ be the linear extrapolation operator that extrapolates $\{(x,y) : |P(x,y)| \leq \delta \cdot \max(|P|)\}$ with a bilinear approximation. Set $\epsilon > 0$ and $\delta > 0$ small and then approximate $P_i^{0,0,0}$ by

$$P_i^{0,0,0}(x,y) \approx \tilde{P}_i^{0,0,0}(x,y) := E_\delta \left(\frac{L(P_i^{0,0,0} f(x,y))}{\tilde{f}^l(x,y) + \epsilon \max_{x,y} |\tilde{f}^l(x,y)|} \right).$$

Theorem 15. *[89] Assume the assumption of Theorem 13.*

Then for $\sup_{u,v} |\hat{f}(u,v) \cdot \mathbb{1}_M(u,v)|$ *small enough,*

$$\left| P_i(x,y) - \frac{L(P_i f(x,y))}{\tilde{f}^l(x,y) + \epsilon \max_{x,y} |\tilde{f}^l(x,y)|} \right| \leq O(\epsilon) + O(\sup_{(u,v)} |\hat{f}(u,v) \cdot \mathbb{1}_M(u,v)|).$$

Proof of Theorem 15. Let

$$\hat{f}_S = \hat{f} \cdot \mathbb{1}_S, \quad \hat{f}_M = \hat{f} \cdot \mathbb{1}_M, \quad \hat{f}_H = \hat{f} \cdot \mathbb{1}_H$$

So $\hat{f} = \hat{f}_S + \hat{f}_M + \hat{f}_H$.

From equation (5.1)

$$L(P_i f)^2 = (P_i f_S)^2 + (\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]}))^2$$
$$+2(P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))$$

Next, from equation (5.2),

$$(\tilde{f}^l)^2 = f_S^2 + \sum_{i=1}^{n}[(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 + 2(P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))]$$

thus

$$P_i^2(\tilde{f}^l)^2 - P_i^2 f_S^2 = P_i^2 \sum_{i=1}^{n}[(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 + 2(P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))]$$

thus

$$\begin{aligned}
&P_i^2(\tilde{f}^l)^2 + \epsilon^2\ P_i^2 \max_{x,y} |\tilde{f}^l(x,y)|^2 + 2\epsilon\ P_i^2 \tilde{f}^l \max_{x,y} |\tilde{f}^l(x,y)| - P_i^2 f_S^2 + L(P_i f)^2 - L(P_i f)^2 \\
&= \epsilon^2\ P_i^2 \max_{x,y} |\tilde{f}^l(x,y)|^2 + 2\epsilon\ P_i^2 \tilde{f}^l \max_{x,y} |\tilde{f}^l(x,y)| \\
&+ P_i^2 \sum_{i=1}^{n}[(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 + 2(P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))]
\end{aligned}$$

then

$$\begin{aligned}
&P_i^2(\tilde{f}^l)^2 + \epsilon^2\ P_i^2 \max_{x,y} |\tilde{f}^l(x,y)|^2 + 2\epsilon\ P_i^2 \tilde{f}^l \max_{x,y} |\tilde{f}^l(x,y)| - L(P_i f)^2 \\
&= -(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 - 2(P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2})) \\
&+\epsilon^2\ P_i^2 \max_{x,y} |\tilde{f}^l(x,y)|^2 + 2\epsilon\ P_i^2 \tilde{f}^l \max_{x,y} |\tilde{f}^l(x,y)| \\
&+P_i^2 \sum_{i=1}^{n}[(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 + 2(P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))]
\end{aligned}$$

then

$$\begin{aligned}
&P_i^2 - \frac{L(P_i f)^2}{(\tilde{f}^l + \epsilon \max_{x,y} |\tilde{f}^l(x,y)|)^2} \\
&= \frac{-(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 - (P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))}{(\tilde{f}^l + \epsilon \max_{x,y} |\tilde{f}^l(x,y)|)^2} \\
&+\frac{\epsilon^2\ P_i^2 \max_{x,y} |\tilde{f}^l(x,y)|^2 + 2\epsilon\ P_i^2 \tilde{f}^l \max_{x,y} |\tilde{f}^l(x,y)|}{(\tilde{f}^l + \epsilon \max_{x,y} |\tilde{f}^l(x,y)|)^2} \\
&+\frac{P_i^2 \sum_{i=1}^{n}[(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 + (P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))]}{(\tilde{f}^l + \epsilon \max_{x,y} |\tilde{f}^l(x,y)|)^2}.
\end{aligned}$$

Let

$$T_2(f,\epsilon) := \frac{-(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M)\cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 - (P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M)\cdot \mathbb{1}_{[-\alpha,\alpha]^2}))}{(\tilde{f}^l + \epsilon \max_{x,y}|\tilde{f}^l(x,y)|)^2}$$
$$+ \frac{\epsilon^2\ P_i^2 \max_{x,y}|\tilde{f}^l(x,y)|^2 + 2\epsilon\ P_i^2 \tilde{f}^l \max_{x,y}|\tilde{f}^l(x,y)|}{(\tilde{f}^l + \epsilon \max_{x,y}|\tilde{f}^l(x,y)|)^2} \tag{5.6}$$
$$+ \frac{P_i^2 \sum_{i=1}^n [(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M)\cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 + (P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M)\cdot \mathbb{1}_{[-\alpha,\alpha]^2}))]}{(\tilde{f}^l + \epsilon \max_{x,y}|\tilde{f}^l(x,y)|)^2}$$

Rearrange and square root

$$\frac{L(P_i f)}{\tilde{f}^l + \epsilon \max_{x,y}|\tilde{f}^l(x,y)|} = \sqrt{P_i - T_2(f,\epsilon)}$$

$$= P_i - T_2(f,\epsilon)\cdot \xi^{-1/2}/2$$

for some

$$|\xi - P_i| < |T_2(f,\epsilon)|$$

and $0 < \xi$. For f_M and ϵ small enough,

$$P_i - T_2(f,\epsilon) > \gamma/2$$

Then

$$|P_i - \frac{L(P_i f)}{\tilde{f}^l + \epsilon \max_{x,y}|\tilde{f}^l(x,y)|}| \le |T_2(f,\epsilon)\cdot \xi^{-1/2}/2| \le \frac{|T_2(f,\epsilon)|}{2\sqrt{\gamma/2}}$$

So

$$\left| P_i - \frac{L(P_i f)}{\tilde{f}^l + \epsilon \max_{x,y}|\tilde{f}^l(x,y)|} \right| \le O(\epsilon) + O(\sup|\hat{f}_M|).$$

□

Remark. *For $\epsilon > 0$, the approximation*

$$P_i(x,y) \approx \tilde{P}_i(x,y) := \frac{L(P_i(x,y)f(x,y))}{\tilde{f}^l(x,y) + \epsilon \max_{x,y}|\tilde{f}^l(x,y)|}$$

decreases in accuracy as $f(x,y) \to 0$ since $\tilde{P}_i(x,y) \to 0$.

Remark. *Now, let $E_\delta(g)$ be the extrapolation operator that linearly extrapolates at points (x, y) where $|g(x, y)| < \delta \max |g|$. Then the approximation*

$$P_i(x, y) \approx \tilde{P}_i(x, y) := E_\delta \left(\frac{L(P_i f(x, y))}{\tilde{f}^l(x, y) + \epsilon \max_{x,y} |\tilde{f}^l(x, y)|} \right)$$

is robust to small values of f since small values of $\tilde{P}_i$ are replaced with the linear extrapolation of the higher accuracy approximations.

The approximation $\tilde{P}_i^{0,0,0}(x, y)$ is reasonable because the profiles are very smooth. Then this implies that the high frequencies of f in the numerator get removed by L and then the low frequencies of f in the numerator get removed by division with approximate low frequencies of f denoted $\tilde{f}^l$ where ϵ is added to avoid division by 0 and increase robustness. Then

$$P_i^{\theta,x_0,y_0}(x, y) \approx \tilde{P}_i^{\theta,x_0,y_0}(x, y) = \tilde{P}_i^{0,0,0}(R_\theta(x, y) + (x_0, y_0)).$$

So now we have the matrices $\tilde{f}^l$ and $\tilde{P}_i^{\theta,x_0,y_0}$. Take the product and use the approximation $P_i^{\theta,x_0,y_0} f \approx \tilde{P}_i^{\theta,x_0,y_0} \tilde{f}^l$ to get the calibration data in the low frequencies.

Theorem 16. *[89] Assume the assumption of Theorem 13.*

Then for $\sup_{u,v} |\hat{f}(u, v) \cdot \mathbb{1}_M(u, v)|$ small enough,

$$|L(P_i^{\theta,x_0,y_0} f) - \tilde{P}_i^{\theta,x_0,y_0} \tilde{f}^l| \leq O(\epsilon) + O(\sup_{u,v} |\hat{f}(u, v) \cdot \mathbb{1}_M(u, v)|)$$

Proof of Theorem 16. We bound $|L(P_i f) - \tilde{P}_i \tilde{f}^l|$. By equation (5.5) and (5.6)

$$\tilde{P}_i^2(\tilde{f}^l)^2 = (P_i^2 - T_2(f, \epsilon)) \cdot (L(f)^2 + T_1(f))$$

$$= P_i^2 L(f)^2 + P_i^2 T_1(f) - T_2(f, \epsilon) L(f)^2 - T_2(f, \epsilon) T_1(f)$$

$$= P_i^2(f_S + \mathcal{F}^{-1}(\hat{f} \cdot \mathbb{1}_{[-\alpha,\alpha]^2 \setminus S}))^2 + P_i^2 T_1(f) - T_2(f,\epsilon)L(f)^2 - T_2(f,\epsilon)T_1(f)$$

$$= P_i^2 f_S^2 + P_i^2 \mathcal{F}^{-1}(\hat{f} \cdot \mathbb{1}_{[-\alpha,\alpha]^2 \setminus S})^2 + P_i^2 f_S \mathcal{F}^{-1}(\hat{f} \cdot \mathbb{1}_{[-\alpha,\alpha]^2 \setminus S}) + P_i^2 T_1(f) - T_2(f,\epsilon)L(f)^2 - T_2(f,\epsilon)T_1(f).$$

Also, from equation (5.1)

$$L(P_i f)^2 = (P_i f_S)^2 + (\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 + (P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2})))^2.$$

Let

$$T_3(f,\epsilon) := L(P_i f)^2 - \tilde{P}_i^2(\tilde{f}^l)^2$$

$$= (\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2}))^2 + (P_i f_S)(\mathcal{F}^{-1}((\hat{P}_i * \hat{f}_M) \cdot \mathbb{1}_{[-\alpha,\alpha]^2})))^2$$

$$- P_i^2 \mathcal{F}^{-1}(\hat{f} \cdot \mathbb{1}_{[-\alpha,\alpha]^2 \setminus S})^2 - P_i^2 f_S \mathcal{F}^{-1}(\hat{f} \cdot \mathbb{1}_{[-\alpha,\alpha]^2 \setminus S}) - P_i^2 T_1(f) + T_2(f,\epsilon)L(f)^2 + T_2(f,\epsilon)T_1(f).$$

Then

$$\tilde{P}_i \tilde{f}^l = \sqrt{L(P_i f)^2 - T_3(f,\epsilon)} = L(P_i f) - T_3(f,\epsilon) \cdot \xi^{-1/2}/2$$

for some $|\xi - L(P_i f)^2| < |T_3(f,\epsilon)|$ and $0 < \xi$.

For f_M and ϵ small enough, $L(P_i f)^2 - T_3(f,\epsilon) > L(P_i f)^2/2$. Then

$$|L(P_i f) - \tilde{P}_i(\tilde{f}^l)| \leq |T_3(f,\epsilon) \cdot \xi^{-1/2}/2| \leq \frac{|T_3(f,\epsilon)|}{2\sqrt{L(P_i f)^2/2}}$$

$$\leq O(\epsilon) + O(\sup |\hat{f}_M|).$$

□

The frequencies in the calibration region of this approximation are used to compute the new GRAPPA kernel and then reconstruct the image, see pseudocode in Algorithm 11.

Algorithm 11: MGRAPPA

Input:
n : measurement resolution
k : number of coils
$Pf \in \mathbb{R}^{n\times n\times k}$: measurement
θ : theta rotation
T : (x,y)-translation
ϵ, δ : regularization parameters

Output:
$R \in \mathbb{R}^{n\times n}$: Image Reconstruction

Begin:
for *i=1...k* **do**
| LPf(i,:,:) = lowpass(Pf(i,:,:))
end
f = sum(abs(LPf(:,:,:)),1)
for *i=1...k* **do**
| P0(i,:,:) = E(LPf(i,:,:)/(f+epsilon*max(f)),delta)
end
for *i=1...k* **do**
| P(i,:,:) = translate(rotate(P0(i,:,:),θ),T)
end
R = Grappa(P.*f) // Entry-wise product

5.1 Simulation

Simulation k-space data was synthesized based on coil sensitivity maps calculated from a phantom experiment and clinical 3-D MP-RAGE coil-combined brain images previously acquired. The phantom experiment was conducted on a 3T Prisma scanner (Siemens, Erlangen, Germany) using a 20-channel head coil and a spherical water phantom (3-D MP-RAGE; FOV = 240×240mm^2, resolution = 0.9×0.9×0.9mm^3, 256×256×256, 256 slices, TR/TE/TI = 2600/2.29/1350ms, flip angle 8°). The human MP-RAGE brain images were acquired on a 3T TrioTim scanner (Siemens) (FOV 256×256mm^2, resolution 1×1×1mm^3, 256×256×160, 160 slices, TR/TE/TI = 2600/4.47/1000ms, flip angle 12°).

When rotation occurs with PMC, the apparent effect is inverse rotation of coil profiles by

the same amount. In this experiment, we use a brain scan as the signal and we use scans of a disk as profiles. Simulation parameters were: 20 coils, full resolution 256x256, calibration data 24*2 lines and $\epsilon = .05$. We simulate rotation of the profiles by calculation using various θ angles and then normalize the reconstructions in sup norm. We plot reconstruction errors for GRAPPA and MGRAPPA, see Fig. 5.7. Error is L2 norm of the residual images; see Fig. 5.6.

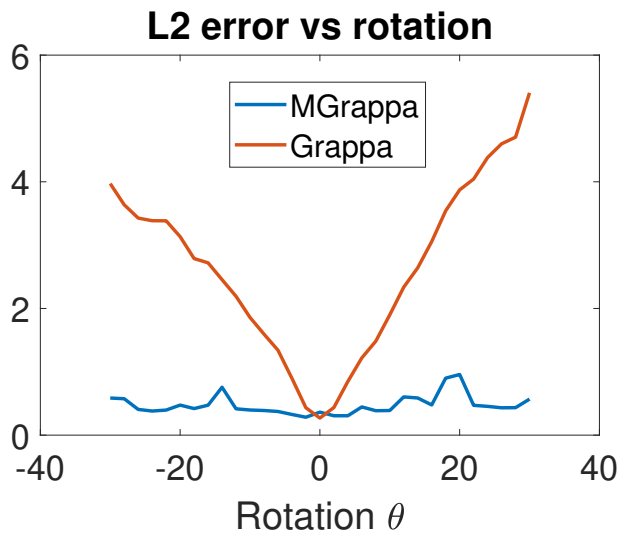


Figure 5.6: Simulated in-plane rotation θ post GRAPPA calibration with prospective motion correction and then plotted reconstruction L2 error of residual with MGRAPPA and GRAPPA. Reconstruction errors increased almost linearly with the rotation angle for conventional GRAPPA, but were essentially flat with MGRAPPA.

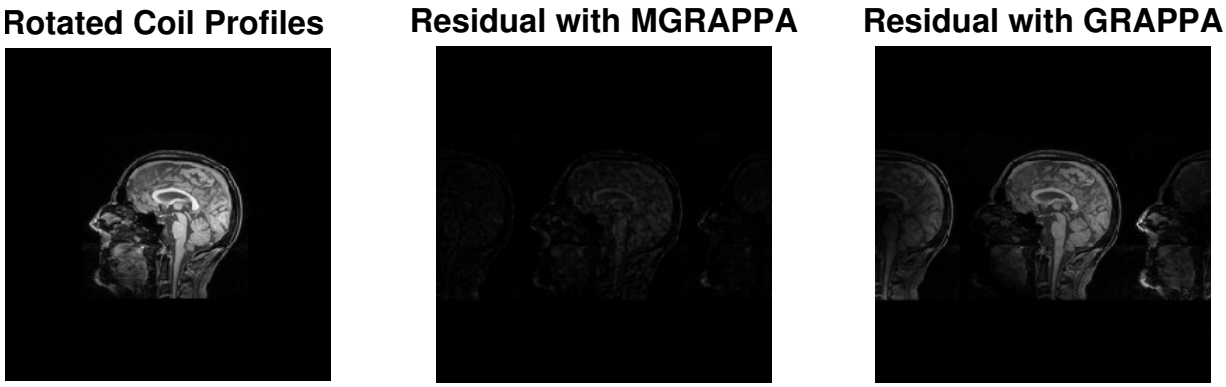


Figure 5.7: Left: Subject simulated 20 degree in-plane rotation post GRAPPA calibration with prospective motion correction. Center: Residual/error of MGRAPPA reconstruction of Subject. Right: Residual/error of GRAPPA reconstruction of Subject.

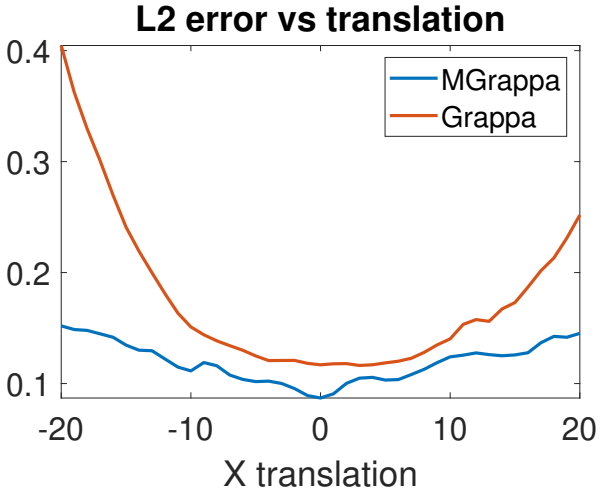


Figure 5.8: Simulated in-plane translation along X axis post GRAPPA calibration with prospective motion correction and then plotted reconstruction L2 error of residual with MGRAPPA and GRAPPA. Reconstruction errors increased almost linearly with the rotation angle for conventional GRAPPA, but were essentially flat with MGRAPPA.

5.2 In Vivo Experiment (Real Data)

Brain MRI was performed in a healthy volunteer (after obtaining consent) who was trained to rotate the head by 7° to the left upon instruction in the middle of the scan. This experiment was conducted on a 3T Prisma with a 20-channel head coil and MP-RAGE sequence with FOV $240\times240mm^2$, resolution $1.9\times1.9\times1mm^3$, 128×128×224, 224 slices, TR/TE/TI=2300/2.29/900ms, flip angle 8°.

We collected full scans, 128x128, of a subject in 2 poses. We used the GRAPPA calibration data, 24*2 lines, from the first pose to reconstruct the scan from the second pose (subsample every other line and then reconstruct), both with and without MGRAPPA correction ($\epsilon = .05$). The L2-norm error decreases 41% with MGRAPPA, see Fig. 5.9.

While the reconstruction error with GRAPPA increases markedly with rotations in an almost linear fashion, MGRAPPA uses approximations to mitigate this and results in a stable method with consistently low errors. In vivo, this appears as less ringing and blurring in the brain image We conclude that our method has great potential to reduce re-scanning, and improve image quality in the presence of motion.

5.3 Open Problems

- Above we used a rectangular sampling scheme, but what is the provably optimal sampling scheme?
- Under statistical noise, is MGRAPPA an unbiased estimator?

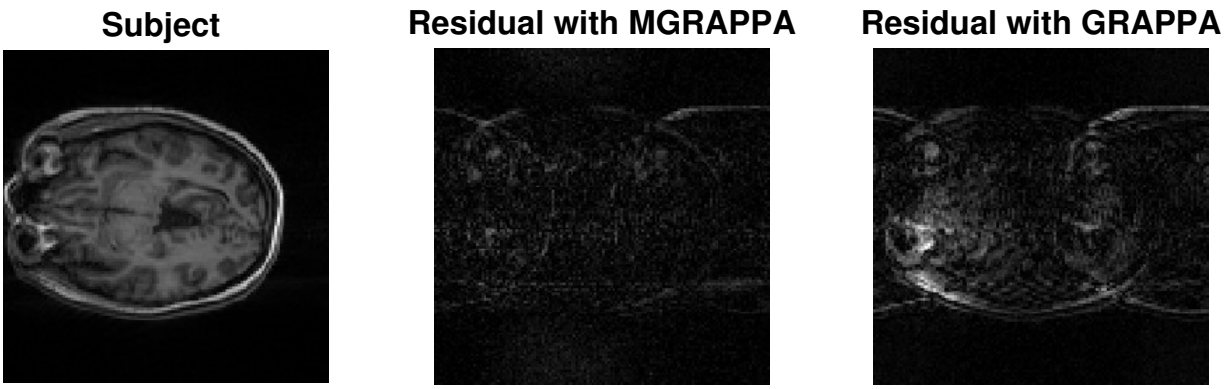


Figure 5.9: Left: Subject. Center: Residual/error of MGRAPPA reconstruction of Subject. Right: Residual/error of GRAPPA reconstruction of Subject. The motion is rotation by 9 degrees and translation by 5 and 1 pixels in x and y directions respectively. GRAPPA reconstruction, but not MGRAPPA, showed a clear ghosting artifact. Accordingly, the L2-norm error decreased 41% with MGRAPPA.

Chapter 6

Persistent Homology Computation Theory

Topology originated in the study of holes. Topology is now defined as the collection of open sets associated to a space. By treating the holes of a space as a basis of a module over a ring (or vector space over a field), we can work with simple objects that represent complicated topological spaces. After defining this carefully, we call this the homology, see [39, 47, 80]. What also comes out is that there are holes of different dimension. Essentially, the k^{th} homology of a space is the vector space on the basis of k^{th} dimensional holes. In applications, we often care about clustering data or detecting bifurcations (or branching) in data. The 0^{th} homology describes the cluster structure. This also corresponds to the bifurcation structure when computed locally. However, given mere data points, the topology is a trivial discrete topology. We need to make some assumptions to reconstruct a meaningful topology of the underlying space of the data.

Let us introduce the Vietoris-Ripps (VR_ϵ) complex from [48]. The VR_ϵ complex takes a

small positive number $\epsilon \in \mathbb{R}$ and assumes that data points with pairwise distance less than ϵ are connected by an edge, creating a graph. A graph is a dimension 1 simplicial complex. We will only consider dimension 1 VR_ϵ complexes because higher dimensions don't affect the 0^{th} homology. If one desires to compute higher dimension VR_ϵ complexes, that is simply done by applying the recursive rule that there will be a k-simplex on some k points if and only if every j-subset of these points has a j-simplex for every $k > j > 1$. What is missing here is what ϵ to use. The resulting homology drastically changes as ϵ is changed. By essentially plotting the homology over ϵ then we can get an idea of the best ϵ.

To define this carefully, instead of a plot we'll define intervals corresponding to the basis elements of the homology. The starting value of each interval will be the smallest ϵ that results in the corresponding homology basis element. The ending value of each interval will be the largest ϵ that results in the corresponding homology basis element. When ϵ is near 0, the VR_ϵ graph has a vertex for each point but no edges. As ϵ increases, many edges are added based on how the data points are distributed. Then as ϵ increases more, the edge additions tapers off. The edge additions will remove basis elements from the homology when the edge connects two disconnected subgraphs. With these assumptions, there will be many short intervals and few long intervals. The long intervals will correspond to the topology of the space. These intervals, called "barcodes", are the persistent homology of the space. There is an algorithm to compute the persistent homology or barcodes, which we describe next.

6.1 Barcode Algorithm

We explain the barcode algorithm for the 0^{th} persistent homology, see [17, 21, 37]. We start with a list of N data points, X, each in d dimensional Euclidean space, $\mathbb{R}^d$. Then we calculate the distance between every pair of points, which corresponds to the ϵ at which an edge is added between the pair in VR_ϵ. Take the list of distances, E, and sort it in increasing order. Remove duplicate elements in E and call it D. We build a matrix, M, with a column for each edge and a row for each vertex in VR_∞ (the complete graph). Set $M(i, j)$ to t a when i is a vertex of edge j and where a is the index of edge j in D which ranges from 1 to the length of D. All other entries of M are set to 0. We make the entries to be polynomials of variable t with coefficients restricted to 0 or 1. The next step is to column reduce this matrix so that what remains is lower triangular. Then the remaining nonzero diagonal entries, say t^b, correspond to the barcode or interval $(0, b)$. With this list of intervals, the algorithm is complete.

6.1.1 CPU Parallelization

Parallelization is highly desirable for the barcode algorithm. The computational complexity of this algorithms is first $O(N^2)$ for the pairwise distance computation, actually over a factor of 2 due to the distance symmetry. Then plus $O(N^2 \log(N^2))$ for sorting the edges. Then plus $O(N^2)$ to compute matrix M. Then plus $O(N \cdot N^2 \cdot N) = O(N^4)$ to reduce, with pivoting, the N by N^2 matrix M. Then plus $O(N)$ to collect the barcode. Altogether this is $O(N^4)$. Even if N is small say $N = 1000$ data points, the time requirement is approximately 10^{12} in time units proportional to the CPU clock rate, a huge amount of time.

Parallelization can leverage multiple processors together to speed up codes and algorithms. We use OpenMP parallelization in C++ to experiment with improving the performance of the barcode algorithm. To use OpenMP we add "for" pragmas before the for loops. In the matrix reduction code, the outer loop is not clearly parallelizable so we put the pragma inside the outer loop where there are parallelizable for loops.

We implement the algorithm in C++ with OpenMP on a 2 core machine running Ubuntu. We use randomly uniformly distributed pairs of numbers in (0,1) for our data. We compile with g++ with -o3 highest compiler optimization since we want to see if the best performance can be improved with parallelization. We note that with -o0 lowest compiler optimization, we get very similar results. We plot averages of 10 runs. In the best case, the run time would be divided by 2 when going from 1 thread to 2 threads. We plot the experiment in Fig. 6.1.

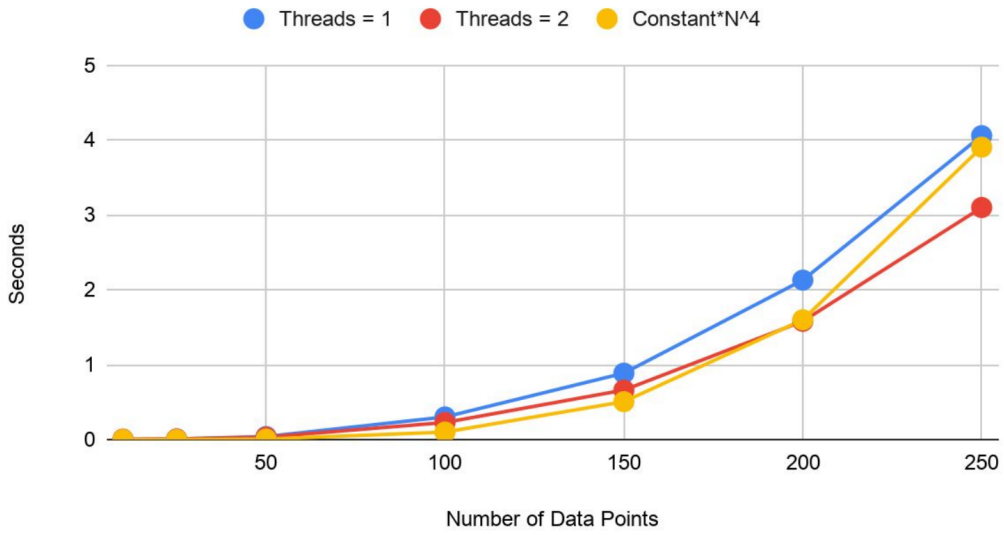


Figure 6.1: Run time to compute the 0^{th} persistent homology versus number of data points using 1 or 2 threads and compared with polynomial growth.

We get an increasing performance gain with the number of data points. The performance gain is less than double however because the code is not perfectly parallel and due to the

thread overhead cost. As for how the run time increases with the number of data points, N, we expect $O(N^4)$ from our calculation above. Even with parallelization and dividing the time by 2, we still expect $O(N^4)$. We plot N^4 times a constant above and see that our experiments are not far off from N^4 though they only need converge as N approaches infinity. Since we use a 2 core machine, more than 2 threads should not give a performance increase. We compare run times with the number of threads greater than 2. We plot the experiment in Fig. 6.2. We see that more threads beyond the number of processing units decreases performance. This is because threads have an overhead cost. Since a threading point is inside of a loop in the matrix reduction code, the thread overhead cost is multiplied by the number of data points, becoming significant.

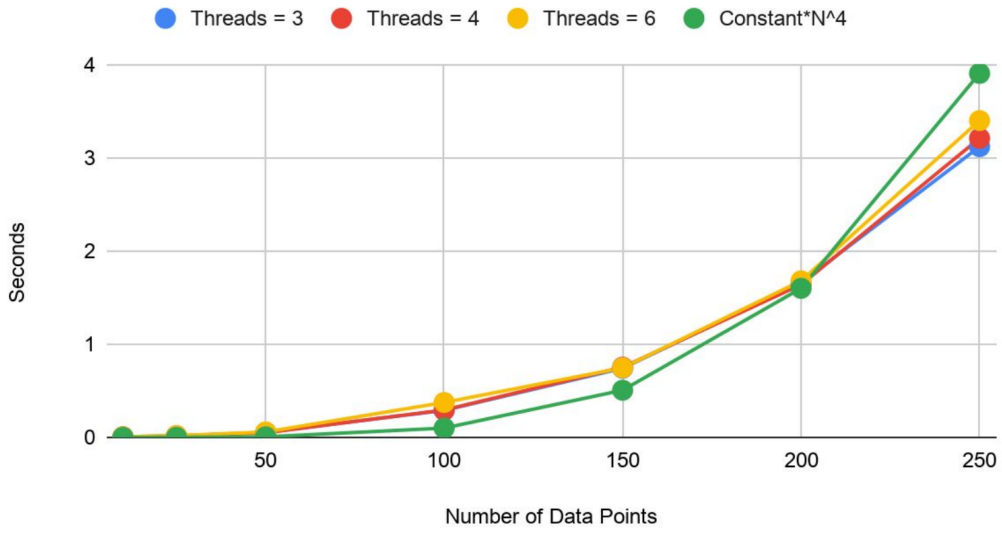


Figure 6.2: Run time to compute the 0^{th} persistent homology versus number of data points using 3, 4, or 6 threads and compared with polynomial growth.

We have shown a large performance increase computing barcodes by using parallelization on a dual core machine. We got up to about 1.75 fold performance increase which may approach the limit of 2 fold increase for a large enough number of data points. We also

showed that the thread overhead cost can accumulate, destroying performance gains but dependent on the implementation. Repeating this work for the higher persistence homology computations won't affect much other than increasing the amount of computation.

6.1.2 GPU Parallelization

First, we use GPU (graphics processing unit) parallelization to compute the 0^{th} Persistent Homology. By mass parallelization, we analytically and empirically decrease the run time scaling from $O(N^4)$ to $O(N^33)$ and even $O(N^2)$ where N is the number of data points, for a large enough GPU. Next, we analytically show run time scaling $O(N)$ for an even larger GPU [86].

GPU (graphics processing unit) usage for data analysis has become possible and popular over the past decade. For example, [72, 104, 105] use GPUs for topological data analysis. As these papers show, putting the GPU's thousands of cores to use in parallel can yield higher performance, that is, lower run times compared to a CPU (central processing unit). As a comparison, consider that an Intel Core i7 980 XE can compute 109 gigaFLOPS [112] (floating point operations) while an NVIDIA Tesla P100 can compute 10 teraFLOPS [46]. This is achievable because even though the GPU's clock rate is only about a third of the CPU's, the GPU has thousands of cores compared to the above CPU's 6 cores. To get an idea of the parallelization of this GPU, we can compute the ratio of FLOPS over clock rate, so 10 teraFLOPS / 1 Ghz = 10^3 operations in parallel [46]. This compares to the above CPU's 109 gigaFLOPS / 3 Ghz = 36 operations in parallel which makes sense given about 6 arithmetic logic units (ALUs) per core.

As explained above, the computational complexity of computing the 0^{th} persistent homology is $O(N^4)$ where N is the number of data points [86]. This is caused by reducing the boundary matrix of size $N \times N(N-1)/2$. Now, as the matrix reduction iterates down the matrix diagonal, each step is easily parallelizable in constant time, ignoring pivoting. This reduction is then a $O(N)$ operation. So can we experimentally observe total run time growth of $O(N)$?

We use a Red Hat Linux server with an NVIDIA Tesla P100 16 GB GPU. We generate N points of uniformly random data on $[0,1] \times [0,1]$ and then we compute the 0^{th} persistent homology as in [86]. We use the GPU to compute the steps of the algorithm in parallel, that is, in constant time. Steps:

1. Compute N(N-1)/2 pairwise distances, that is the edges.

2. Sort the edges.

3. Create the boundary matrix.

4. For each diagonal entry of the matrix: Reduction step.

5. For each diagonal entry of the matrix: Get the barcode interval.

Then we plot the run time using the GPU vs sequential CPU code without the GPU, see Fig. 6.3. The experiments are averages of 10 runs. We notice that the run time of the CPU (no GPU) code grows with $O(N^4)$ as the theory predicts. The run time of the GPU code grows with $O(N^2)$ from N=50 to 150. Then the run time of the GPU code grows with $O(N^3)$ from N=200 to 700. A note on pivoting, pivoting, to the extent that it is happening, may be thought to be affecting the theory and experiments. Experimentally, even when pivoting

is disabled, the result is very similar. As for the theory, finding the pivots could be done with the GPU in logarithmic time by doing a binary search via sums of row slices. Another note on sorting the N(N-1)/2 edges by length, sorting on the GPU in parallel, for a large enough GPU, can be done in $O(\log(N(N-1)/2))$ run time [79]. Our implementation uses the Thrust library for sorting on the GPU and we observe it takes negligible run time.

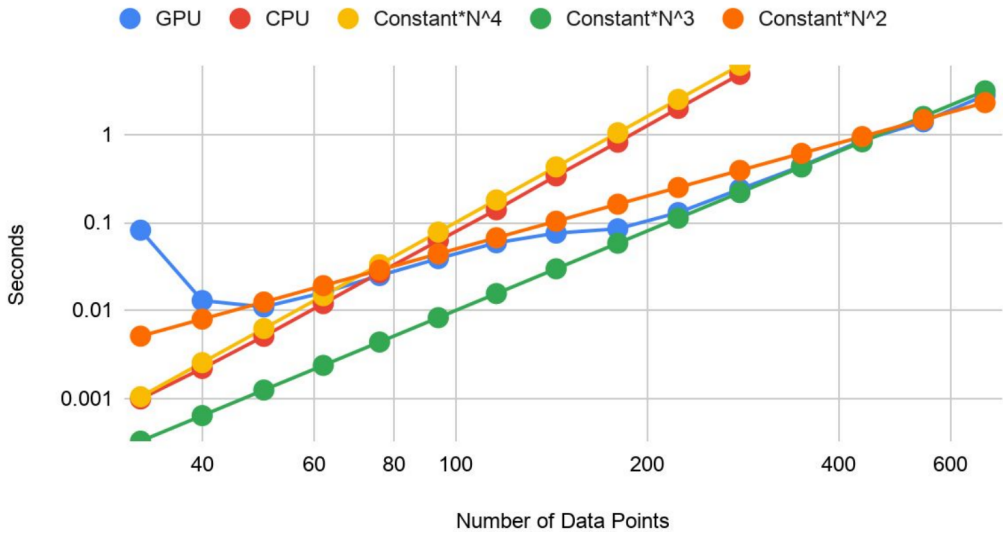


Figure 6.3: Run time to compute the 0^{th} persistent homology versus number of data points using CPU or GPU and compared with polynomial growth.

The reason that we didn't get $O(N)$ run time growth is because we made a crucial assumption above. We assumed that the GPU could do $N \times N(N-1)/2$ operations in parallel. Using our approximate calculation above of 10^3, that means $N < 28$. At those small N values, the memory and initialization times can swamp the calculation. However, if $N < 10^3$ or if $N(N-1)/2 < 10^3$ then we can do column or row, respectively, operations in parallel and then decrease the run time to $O(N^3)$ or $O(N^2)$, respectively. We calculate those thresholds and see $N = 150$ and $N = 10^3$. This matches the data where we see $O(N^2)$ run time growth up to N=150 and then after we see $O(N^3)$ run time growth. We did not

run N beyond 700 due to limited time. Of course for N large enough the run time growth must be that of the computational complexity which is $O(N^4)$. Technically, if the computing resources are finite and constant, then they will not change the computational complexity as N grows very large.

We have shown analytically and experimentally that with a large enough GPU, we can decrease the run time growth of computing the 0^{th} persistent homology from $O(N^4)$ to $O(N^3)$ and even $O(N^2)$. Analytically, with many large GPUs, $O(N)$ run time growth is possible. However, we've ignored memory transfers and latencies which could be researched in future work. Future work also includes the straight forward extension to the higher order homology groups.

6.2 Natural Language Processing Application

Natural language processing (NLP) seeks insights from data in form of a natural language such as English. We will see how to analyze natural language data with persistent homology [85]. We often want to understand the shape or topology of any dataset we are given. We may think there is a linear relationship between the variables and fit a linear model. We may believe that the data comes from a sphere and visualize them to test this theory. Often, we do not know what the shape of the data is, but believe the data has been sampled from some underlying surface or manifold lesser in dimension than the ambient space. The shape of this surface is interesting, but difficult to analyze so we focus on topological features which are much simpler. For example we check if the surface has holes or even multiple connected components which will correspond to clusters of the data.

To formalize this question, suppose you are given some (finite) data X which you believe comes from some surface, or low dimensional manifold, $\mathbb{X}$. The critical question is whether we can infer the shape (homology: number of holes, components, etc.) of $\mathbb{X}$ from the given data X. Consider the data to the right[1]. Intuitively we see that this data comes from noisily sampling a circle because there is some hole in the middle of the data. But how would one algorithmically detect this for any given data? The most common way to do this is to use a concept called persistence. The core idea of persistence is to see how the union of balls around each point becomes connected as the balls grow in radius.

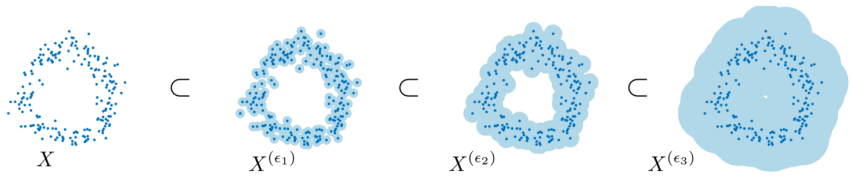


Figure 6.4: Illustration of Topology. Image credit to [77]

We see that as we grow an ϵ-ball around each point, the union of balls becomes more connected until we have all balls connected and 1 hole in the middle of the union. In persistent homology, we examine how these connections evolve overtime (time being the ball radius increasing). For the different discriminators of different shapes (connected components, holes, etc), we can observe their birth and death as we increase ϵ. By focusing on those discriminators which have a long life, i.e., persist, we can begin to understand the shape of our data.

[1]Figures from reference [77]

This field has been studied for a long time, dating back to the 1940s ([69], [35], and [91]). These works focus on the theory of topology, but they were restricted in their applicability due to their lack of computation. Then [28] introduced a fast algorithm to compute a surface's topology, and this led to the focus of computational techniques in the field of topology. After the publication of this work, many subsequent work was focused on advancing the theory and computability of the shape of surfaces. For an abstract overview of these works see [60]. For a history of persistent homology in topological data analysis (TDA) see [77]. For more mathematical and technical descriptions of the field see [19, 27, 38].

There are only six published works at the intersection of NLP and TDA [66, 97, 98, 106, 109, 120], at time of writing. The first concrete example of a successful application of TDA to NLP comes from [109] which demonstrates the difficulties and possibilities for computational topology to analyze the similarities within a collection of text documents. [106] argues for applicability of persistent homology to lexical analysis using word embeddings. This paper aims towards the same goal as ours, but does so using a slightly different TDA method, which we believe we can improve upon. [98] applies TDA to entailment, with an improvement of accuracy over the baseline without persistence.

Word sense disambiguation (WSD) was first framed as a computational task in the 1940s with Zipf's power-law theory. Since then, the types of solutions to WSD can be binned into three categories: knowledge-based, supervised, and unsupervised. An example of the knowledge-based approach is [67] where semantic similarity is used to measure distances between words which theoretically "measure" how much words are semantically similar. The approach taken in this example is different from our unsupervised approach, but the end goal is the same. [67] support their claim of the appropriateness of (their specific)

similarity measures by using real time implicit feedback from user. An example of the supervised approach is [18] which uses a combination of different models: Naive Bayes, maximum entropy model, boosting, and kernel-PCA. This result [85] is the first of its kind to provide evidence that WSD can be used to improve the performance of statistical machine translation (SMT) tasks. Like the previous example, the goal is the same as ours but the approach is different. [18] justify their claim using experiments comparing SMT against SMT + WSD across a variety of tasks and a variety of performance measures. The final approach is an unsupervised approach, like [100]. These authors posit that graph-based centrality measures for word sense disambiguation can capture the necessary information. This approach is similar to ours in that there is an assumption that the geometry of a space carries some information about word senses. However, their work uses different similarity measures directly, while ours will use inferred measures through word embeddings.

We will use training and evaluation datasets from Linguistic Computing Laboratory group at the Sapienza University of Rome. The training data consists of two sense annotated training corpora, SemCor and OMSTI. All senses are annotated with WordNet 3.0. The XML data file contains the following tags: corpus $\rightarrow$ text $\rightarrow$ sentence. Then, each sentences consists of "wf" (non-disambiguated) and "instance" (disambiguated) tags. We will use persistent homology as described above to examine the shape of common word embeddings.

Firstly, we will compute word embeddings with word2vec, GloVe, and fasttext on the datasets SEMEVAL-2013, SemCor, and SemCor+OMSTI. Then, we will compute barcodes to understand the topology, or shape, of the word embedding space. We will analyze the results of this algorithm by finding the topological components that each word belongs to, as output from the barcodes algorithm. We hypothesize that this information will have

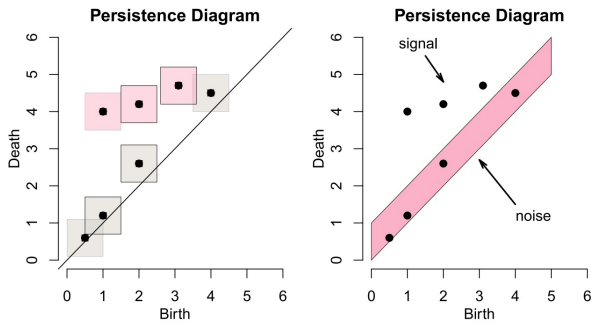


Figure 6.5: Diagram Algorithm Output. Figure from [77].

some signal about the number of word sense for each word. In addition, we will construct pseudowords by concatenating unrelated words, replace them in the corpus and check if the barcode algorithm recovers the senses of the two words. We will use the datasets SEMEVAL-2013, SemCor, and SemCor+OMSTI. We will compute word embeddings for each dataset using all three methods: word2vec, GloVe, and fasttext. This would output a total of nine word embeddings. We will perform this computation on server clusters for mass parallelization.

Recall, that the study of topology examines the surfaces that are generating the data. For instance, we assume that our word embeddings are produced from some underlying manifold that has particular structures of interest. For instance, word embeddings make "similar" words cluster together in the embedding space. We note that the definition of similar is somewhat nebulous. In topology, we call these clusters *connected components* because if we were to connect each of the points/word embeddings to any other point that is say ϵ away, then they would form one connected piece. This point can be seen in Fig. 6.4. Recall that persistent homology is the study of the evolution of these topological features as we increase ϵ.

The output of the Barcode algorithm will be a diagram like in Fig. 6.5. This diagram will tell us which are topological features of the word embedding space that are meaningful. In Fig. 6.5 they are the three signals on the right which fall above the noise-thresholded diagonal. With the topological information output from the barcode algorithm, we can look at words that have multiple senses. The essential question we ask will be, if we remove this word/point from the embedding, does the topology change. To think of this more concretely, consider the word "bank" which has a financial and a alluvial meaning, each with their own cluster of similar words. However, "bank" falls between them and so if it is removed, the connection between the two clusters will be broken and one connected component will become two. We shall compute these changes for various words to analyze the topological relation to the semantics.

Once we run the above barcode algorithm on the word embeddings for the different datasets and perform the analysis, we will repeat the process but with pseudowords. For a word that has multiple meanings or senses, we can break that word up into its component senses, create new dummy words, and replace the original word with the respective dummy word. For example, if the word "foo" has two meanings, we can create words "foo$1" and "foo$2" and replace "foo" with the respective pseudoword we just created.

Once we have done this, we will replace the words in the corpora, retrain the embeddings, and rerun the barcode algorithm. We will proceed entirely as before with our analysis and draw conclusions about the appropriateness of topological data analysis in word sense disambiguation/induction.

We will compare the number of senses calculated for each word with the ground truth from the annotated datasets SEMEVAL-2013, SemCor, and SemCor+OMSTI to the barcode

algorithm prediction. We shall look at the topological features of words with multiple sense. We will then compute how the topology changes when that word is omitted. The average relative error, comparing word by word, will be a measure of success with a perfect match having average relative error = 0. We will measure the success against the hyperparameters δ, the locality radius, and ϵ, the noise sensitivity. That is if g is the ground truth vector of number of word sense per word with length n, and $\tilde{g}$ is our approximate, then the average relative error is

$$e_{\delta,\epsilon} = \frac{1}{n}\sum_{i=1}^{n} \frac{|(g)_i - (\tilde{g}_{\delta,\epsilon})_i|}{(g)_i}.$$

As stated above, we will also calculate this error on the pseudowords to measure our success there.

The above measures the relative difference in word sense that our barcode algorithm predicts versus what are given as ground truth. This will measure effectively how this particular topological approach worked with recovering word senses. However, we note that even if this measure is large, there might be useful information encoded with the barcode algorithm. Therefore, we will also perform qualitative analyses to derive reasons for the results.

Our baseline results have been computed on the Semcor dataset annotated using WordNet with the word2vec embedding training routine. We define (number of senses = number of death dates that are > (mean + 2 standard deviations)). We compute absolute and relative errors when compared to the ground truth number of senses, see Fig. 6.6.

We plot number of closest words considered vs absolute error, see Fig. 6.7. If we consider very few neighboring words, we have less information than what is required and for a large

Embedding Dimension	Num Neighbors Considered	Relative Error	Absolute Error
500	100	0.9023	2.723
500	50	0.4178	1.660
500	25	0.4114	2.202
100	100	0.8030	2.447
100	50	0.4331	1.745
100	25	0.4348	2.287

Embedding Dimension	Num Neighbors Considered	Relative Error	Absolute Error
1000	200	0.2129	2.907
1000	100	0.4609	6.511
500	200	0.1897	2.6511
500	100	0.4712	6.6279
100	200	0.2068	3.0465
100	100	0.4855	6.5116

Figure 6.6: Table of relative and absolute error for word sense disambiguation versus different embedding dimensions and number of neighbors considered.

number words we have too much irrelevant information. In both cases, error is higher as expected. The minimal error is in the middle, hence our results are coherent with our expectations.

6.3 Open Problems

- Can a provably approximate barcode be calculated in linear time with limited hardware?

- Can the d dimensional barcode be used to decrease the computation of the k dimensional barcode where $d \neq k$?

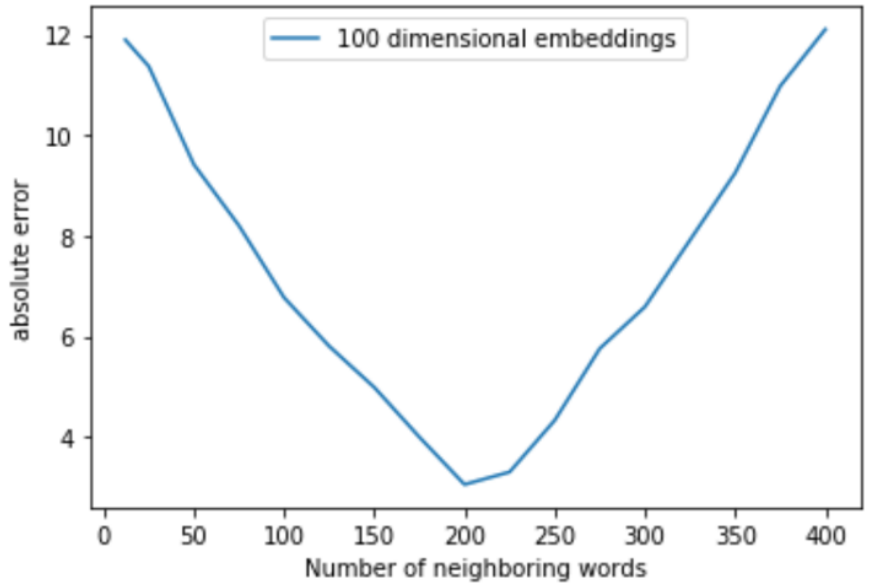


Figure 6.7: Plot of absolute error for word sense disambiguation versus number of neighbors considered, using 100 dimensional word embeddings.

Chapter 7

Geometric Reach

In application and theory, reconstruction of an embedded manifold [58] from samples hinges on the manifold's *curvature* and *reach*. The reach of a manifold is the smallest distance to the *medial axis*. Reach is a function of curvature but contains more information for reconstruction. We show instability, theoretically and practically, in Federer's manifold reach formula [31].

The simplest interpolation method is *linear interpolation* which we discussed in Chapter 2. Linear interpolation is a tried and true method that works for numerous applications. Consider sampling part of a circle and plotting a linear interpolant, Fig. 7.1. As we see the linear interpolant matches the data well in this case, though this is a bit of an artifact. If the samples came from a larger interval, then clearly the linear interpolant would not match the sample. We need to quantify mathematically where and when linear interpolation fails. We can do so by calculating the (extrinsic) curvature [58] of the data distribution, which in this case consists of all points (x, y) on the circle. When the curvature is very high, the linear interpolant completely fails to match the data, as is the case shown in Fig. 7.2.

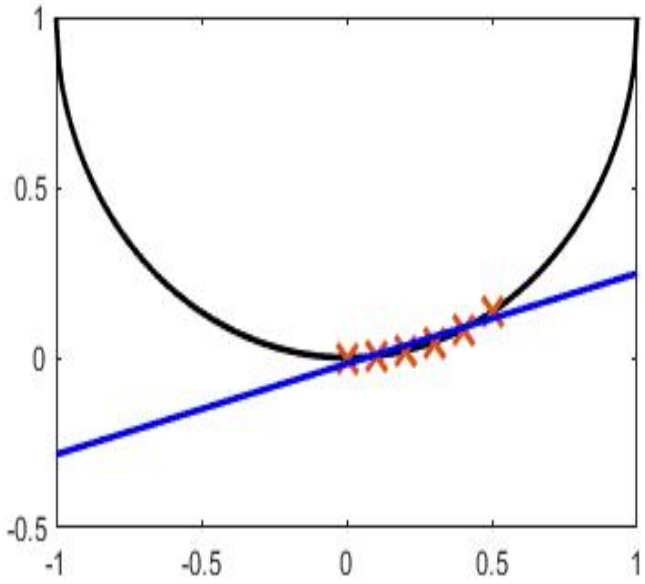


Figure 7.1: In black, half-circle and samples marked with red X. In blue, linear interpolation of samples.

Quantifying and controlling curvature is crucial to properly using linear interpolation in theorems and applications. Because curvature can vary at each point, standard procedure is to require (scalar) curvature to always be below a threshold. This requirement along with a minimal gap requirement gives us a requirement on *reach*. *Reach* is defined as the maximum of a curvature term and the minimal gap between pieces of the data distribution [1]. Now given a *distribution*, we can define the reach τ_D of distribution $D \subset X$ by

Definition 7.0.1.

$$\tau_D = \inf_{x \in X \setminus D} \{d(x,y) : \exists y, z \in D, y \neq z, d(x,y) = d(x,z)\} \cup \{\infty\}$$

where $d : X \times X \to \mathbb{R}_+$ is a distance on X.

In Fig. 7.2, the space X is the plane and the distribution D is a density on the circle.

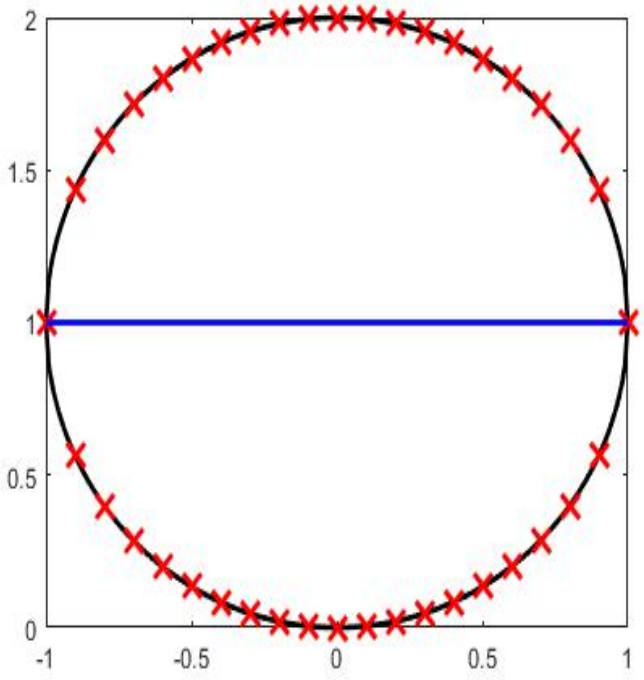


Figure 7.2: In black, circle and samples marked with red X. In blue, linear interpolation of samples.

In the case that the data distribution is a manifold in X, the reach is equal to the minimum of bottleneck distances,

$$\{d(x,y) : x \in X \backslash D, y \in D : \exists z \in D, x \neq z, d(x,y) = d(y,z) = d(x,z)/2\},$$

and a curvature term [1], see Fig. 7.3.

In this section, we will work with data distributions that are smooth manifolds which are smoothly embedded in Euclidean space going forward except where noted. In the case of compact manifolds without boundary in a normed space, Federer [31, Thm. 4.18] showed that the reach of manifold M is

$$\tau_M = \inf_{x \neq y; x,y \in M} \frac{\|y - x\|_2^2}{2d(y - x, T_x M)}$$

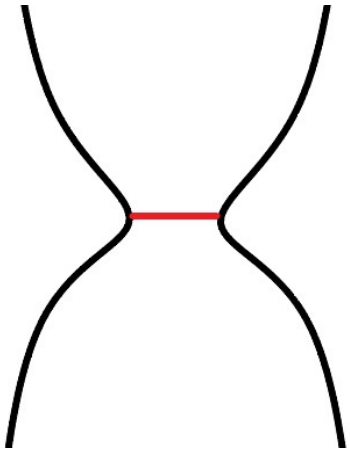


Figure 7.3: Manifold in black and red line showing bottleneck location.

where T_xM is the tangent space of M at x embedded in the ambient space. Many works have utilized this formulation such as [1]. However, there exists an instability which makes an accurate computation almost impossible. Given a set of samples $S \subset M$, let estimate

$$\hat{\tau}_M = \inf_{x \neq y; x,y \in S} \frac{\|y-x\|_2^2}{2d(y-x, T_xM)}.$$

Suppose a sample or tangent space is perturbed as in Fig. 7.4. We plot samples including one that falls outside of the black straight line manifold. We plot a tangent line estimate in blue at the erroneous sample. Then, with the error, the estimate $\hat{\tau}_M$ decreases to a small, inaccurate result. We give the cause of this in the following theorem.

Theorem 17. *For N samples, S, of smoothly embedded manifold $M \subset \mathbb{R}^m$ and some perturbation p of T_xM and $\eta(r) = \inf_\gamma \|proj_{T_xM}(\gamma(r) - x)\|$ where γ is a geodesic and some $0 \leq \xi \leq \|y-x\|$,*

$$\hat{\tau}_M \leq \inf_{x \neq y; x,y \in S} \frac{\|y-x\|}{2\|p\|(\sup \|\gamma''|_x\| + \eta''(\xi)\|y-x\|)}$$

Remark. *Thus with worst case perturbations, for number of samples N large, $\hat{\tau}_M \to 0$ since*

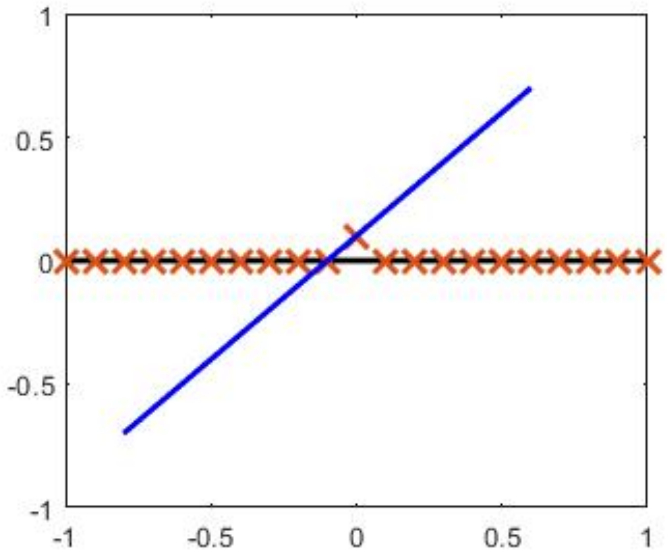


Figure 7.4: Noisy samples in red from manifold in black. Blue line showing inaccurate tangent space.

$\inf_{x\neq y;x,y\in S}\|y-x\|\to 0$ *for i.i.d. samples* S. *We note that a perturbation of the tangent space is equivalent to perturbing a sample since the samples estimate the tangent space.*

Proof of Theorem 17. Let $\mathcal{N}_x(M)$ be the perpendicular space to the tangent space T_xM. There is a vector $v_{\mathcal{N}}\in\mathcal{N}_x(M)$ such that $d(y-x,T_xM)=\langle y-x,v_{\mathcal{N}}\rangle$. Then

$$\hat{\tau}=\inf_{x\neq y}\frac{\|y-x\|^2}{2\langle y-x,v_{\mathcal{N}}\rangle}.$$

Let $p\in T_xM$ and curvature $c_x=\sup\|\gamma''|_x\|$ for $\gamma:\mathbb{R}\to M,\gamma(0)=x,\|\gamma'\|=1,proj_{T_xM}(\gamma'')=0$, that is a geodesic. Let $\eta(r)=\inf_\gamma\|proj_{T_xM}(\gamma(r)-x)\|$ and $0\leq\xi\leq\|y-x\|$.

$$\begin{aligned}\hat{\tau}&=\inf_{x\neq y}\frac{\|y-x\|^2}{2\langle y-x,v_{\mathcal{N}}+p\rangle}\\&=\inf_{x\neq y}\frac{\|y-x\|^2}{2\langle y-x,v_{\mathcal{N}}\rangle+2\langle y-x,p\rangle}\\&=\inf_{x\neq y}\frac{\|y-x\|^2}{2\langle y-x,v_{\mathcal{N}}\rangle+2\langle proj_{T_xM}(y-x),p\rangle+2\langle proj_{\mathcal{N}_xM}(y-x),p\rangle}\end{aligned}$$

$$
\begin{aligned}
&= \inf_{x \neq y} \frac{\|y-x\|^2}{2\langle y-x, v_{\mathcal{N}} \rangle + 2\|proj_{T_xM}(y-x)\|\|p\|} \\
&\leq \inf_{x \neq y} \frac{\|y-x\|^2}{2\langle y-x, v_{\mathcal{N}} \rangle + 2\|p\|\eta(\|y-x\|)} \\
&= \inf_{x \neq y} \frac{\|y-x\|^2}{2\langle y-x, v_{\mathcal{N}} \rangle + 2\|p\|(c_x\|y-x\| + \eta''(\xi)\|y-x\|^2)} \\
&\leq \inf_{x \neq y} \frac{\|y-x\|^2}{2\|p\|(c_x\|y-x\| + \eta''(\xi)\|y-x\|^2)} \\
&= \inf_{x \neq y} \frac{\|y-x\|}{2\|p\|(c_x + \eta''(\xi)\|y-x\|)}.
\end{aligned}
$$

□

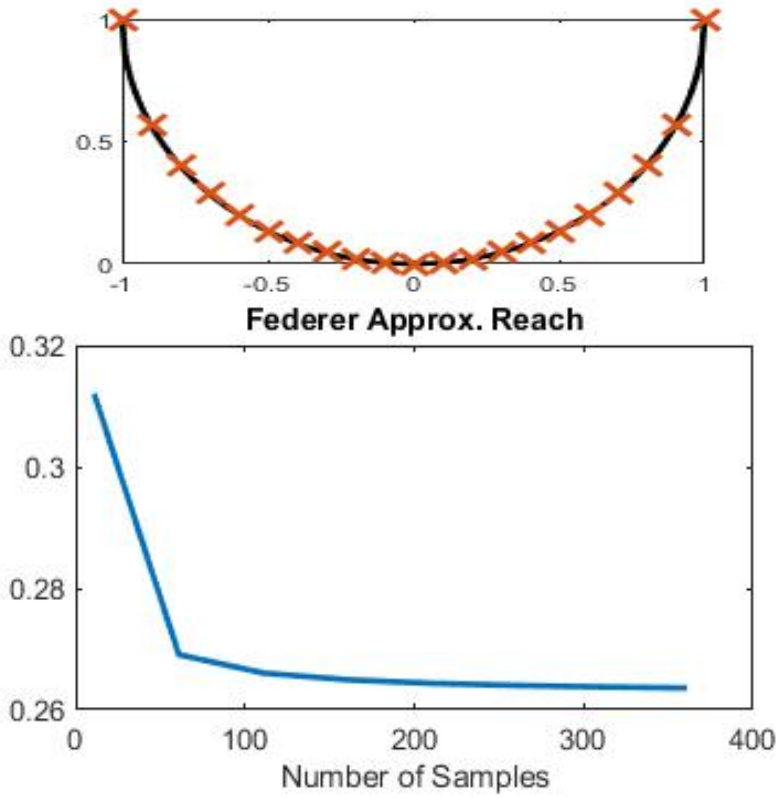


Figure 7.5: Top: Samples as red X from half-circle manifold in black. Bottom: Blue line showing Federer's approx. reach of manifold. Manifold reach is equal to 1.

Next, we show how the error term grows with samples in Fig. 7.5. We plot a semicircle manifold and a set of samples. Then we calculate $\hat{\tau}$ and vary the number of samples.

Typically, more samples leads to a better model, however here neither less data nor more data results in an accurate estimate, since $\hat{\tau} \neq \tau = 1$ in Fig. 7.5. To compensate for this defect, we introduce a combinatorial reach which is a robust multiscale generalization of reach.

7.1 Combinatorial Reach

Consider the simplicial complex $S = \{[v_{i_1}, ..., v_{i_n}]\}_{\{i\}_1^n \in I}$, where I is an index set and v_i are labels of vertices embedded in metric space $(X \subset \mathbb{R}^m, d, \|\cdot\|)$. We always will assume the faces of S are linear in $\mathbb{R}^m$. Let $\{V_k\}_{k\in I}$ be the Voronoi complex of the vertices of S. For each k, the cell is given by

$$V_k = \{x \in X : d(x, v_k) < d(x, v_j)\ \forall j : v_k \neq v_j\}.$$

Using the Voronoi complex, define V_δ as the complex formed by the boundaries of the cells V_k that are δ separated from S. Let

$$V_\delta = \{\sigma \in complex\,(\cup_k\ cl(V_k) - V_k) : d(\sigma, S) > \delta\}$$

where $d(\sigma, S) = \inf_{x\in\sigma, y\in S} d(x, y)$. In Fig. 7.6, we show a simplex S, V_k, and V_δ.

Definition 7.1.1. Define the *combinatorial reach* of simplicial complex S as

$$\tau_S(\delta) := \min\left(d(S, V_\delta),\ \sup_{u,w\in X} d(u, w)\right)$$

which may equal ∞.

Let's take a look at a few examples and their combinatorial reach which we will abbreviate as c-reach. In Fig. 7.7 we see the c-reach of samples from three different spaces. The circles

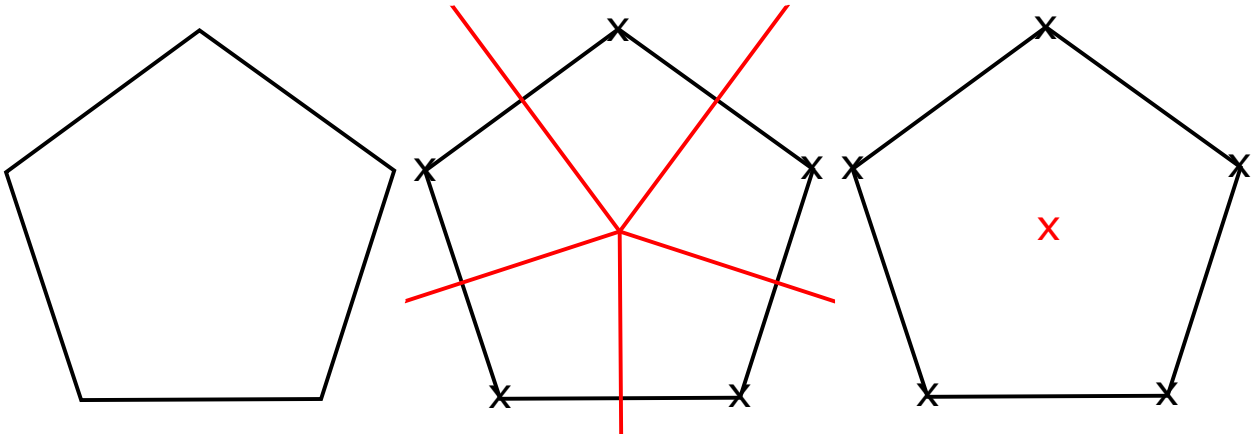


Figure 7.6: Left: We show a simplicial complex. Center: A simplicial complex is in black. The vertices are marked X. The Voronoi complex is in red. Right: The simplicial complex is in black. We show V_δ complex in red for $\delta <$ the circumscribed circle's radius.

c-reach reflects three values. First level on the left is half the distance between samples, then the radius is reflected, and finally ∞ is achieved. Two parallel lines causes a bottleneck. The spaces c-reach starts as half the distance between samples, then reflects the bottleneck distance, and finally is ∞. We have seen that in the fundamental cases, the c-reach contains the curvature and bottleneck distance. How does the c-reach handle a perturbation of a sample on an interval? We plot this in Fig. 7.7 and see that the perturbed or kinked space has a c-reach that starts at half the distance between points and then takes small steps up to ∞. The correct reach of ∞ is contained in the c-reach beyond some level. This improves over the standard reach which is very low and this improves over Federer's formula which gives a low value. A few properties become apparent.

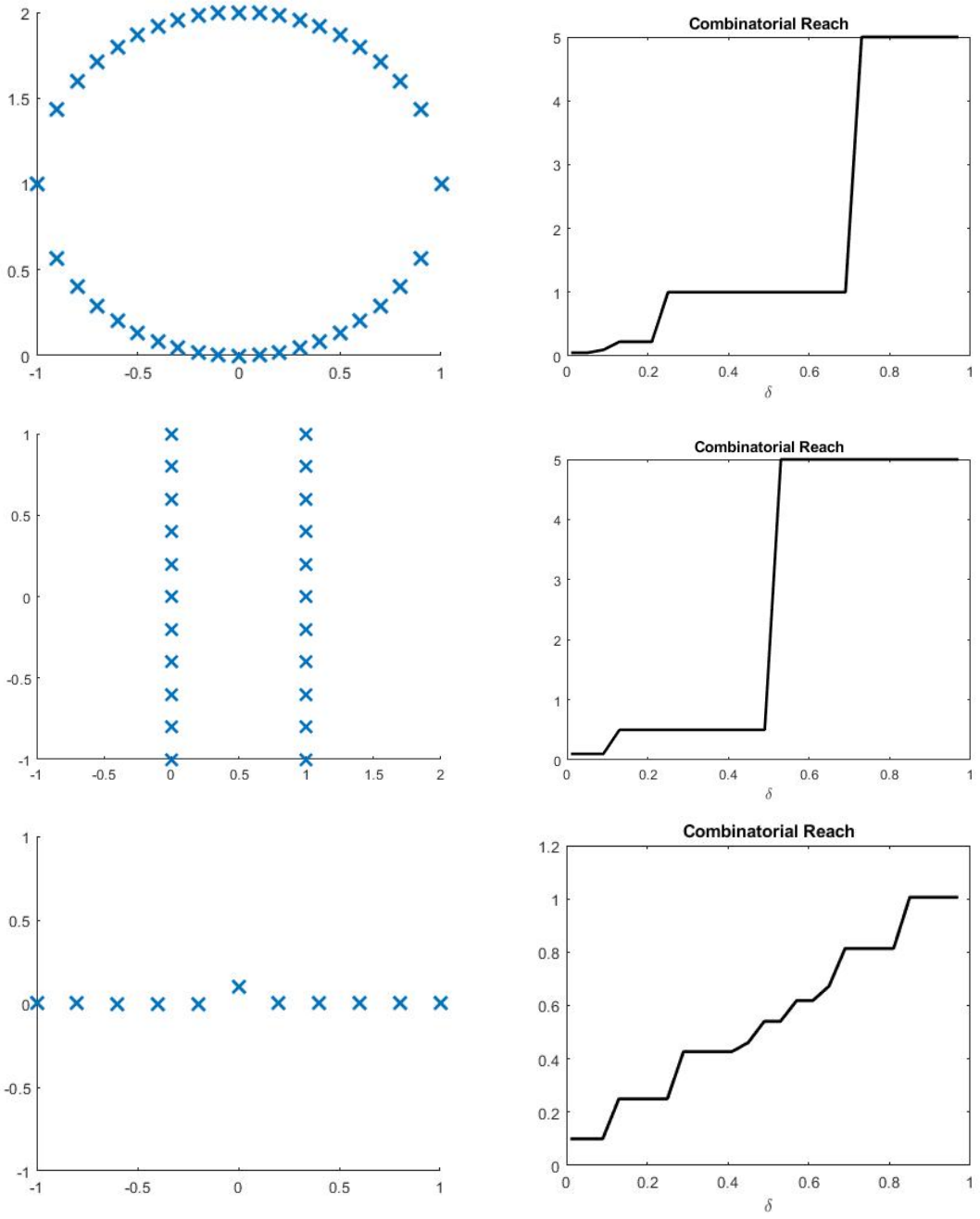


Figure 7.7: Left Top: Samples from a circle marked with blue X. Right Top: Combinatorial reach of circle samples. Left Middle: Samples from parallel lines marked with blue X. Right Middle: Combinatorial reach of parallel lines samples. Left Bottom: Noisy samples from interval marked with blue X. Right Bottom: Combinatorial reach of noisy interval samples.

Proposition 7.1.1. C-reach is a non-negative, monotonically increasing function up to $\sup_{u,w\in X} d(u,w)$ and is then constant.

Remark. *C-reach is a global function in the sense that perturbing one sample can change c-reach at any δ.*

However, in compact ambient spaces, c-reach is Lipschitz continuous, in L^p, to perturbations of vertices. In a compact space $X \subset \mathbb{R}^m$, V_k depends Lipschitz continuously on perturbations of v_j for any k, j in the Hausdorff (HD) measure. V_δ depends Lipschitz continuously on perturbations of v_j for any j. τ_S depends Lipschitz continuously on perturbations of v_j for any j in $\|\cdot\|_{L^p}$. So τ_S depends Lipschitz continuously on perturbations of the vertices of complex S. Adding or deleting interior cells (not a vertex) which is within δ in HD, only changes τ_S below δ.

Theorem 18. *In a compact ambient space $X \subset \mathbb{R}^m$, c-reach of simplicial complex S is Lipschitz continuous, in L^p, to a small enough perturbation $\eta \in \mathbb{R}^m$ of a vertex. Let $S^{(\eta)}$ be complex S with perturbation η. Then*

$$\|\tau_{S^{(\eta)}}(\cdot) - \tau_S(\cdot)\|_{L^p} \leq L|X|^{1/p}\|\eta\|$$

Remark. *A finite sequence of small enough perturbations is also Lipschitz continuous for τ_S via triangle inequality.*

Proof of Theorem 18. Part I. The medial axis of v_k and v_j, medial_axis(v_k, v_j) =

$$A = \{x \in X : (x - \frac{v_k + v_j}{2}) \cdot (v_k - v_j) = 0\}.$$

Let medial_axis$(v_k + \eta, v_j)$=

$$A_\eta = \{x \in X : (x - \frac{v_k + \eta + v_j}{2}) \cdot (v_k + \eta - v_j) = 0\}.$$

Use the Hausdorff distance

$$d(A_0, A_\eta) = \inf\{\epsilon : A_0 \subset \cup_{x\in A_\eta} B(x,\epsilon), A_\eta \subset \cup_{x\in A_0} B(x,\epsilon)\}.$$

We calculate the Lipschitz constant of $d(A_0, A_\eta)$ as a function of η for $\|\eta\| \leq \frac{1}{2}\|\triangle S\|_{\min}$.

Take $x \in A_0$ and $x + w \in A_\eta$ where $\|w\|$ is minimal. Then

$$\frac{w}{\|w\|} = \pm\frac{(v_k + \eta - v_j)}{\|v_k + \eta - v_j\|}$$

Since $x + w \in A_\eta$ then

$$0 = (x + w - \frac{v_k + \eta + v_j}{2}) \cdot (v_k + \eta - v_j)$$

$$= (x - \frac{v_k + \eta + v_j}{2}) \cdot (v_k + \eta - v_j) + w \cdot (v_k + \eta - v_j)$$

$$= (x - \frac{v_k + \eta + v_j}{2}) \cdot (v_k + \eta - v_j) \pm \|w\|\frac{(v_k + \eta - v_j)}{\|v_k + \eta - v_j\|} \cdot (v_k + \eta - v_j)$$

$$= (x - \frac{v_k + \eta + v_j}{2}) \cdot (v_k + \eta - v_j) \pm \|w\|\|v_k + \eta - v_j\|$$

So

$$\|w\| = \frac{\left|(x - \frac{v_k+\eta+v_j}{2}) \cdot (v_k + \eta - v_j)\right|}{\|v_k + \eta - v_j\|}$$

$$= \frac{\left|(x - \frac{v_k+v_j}{2}) \cdot (v_k - v_j) + (x - \frac{v_k+v_j}{2}) \cdot \eta - (\frac{\eta}{2}) \cdot (v_k + \eta - v_j)\right|}{\|v_k + \eta - v_j\|}$$

since $x \in A_0$ and $(x - \frac{v_k+v_j}{2}) \cdot (v_k - v_j) = 0$

$$= \frac{\left|(x - \frac{v_k+v_j}{2}) \cdot \eta - (\frac{\eta}{2}) \cdot (v_k - v_j) - (\frac{\eta}{2}) \cdot \eta\right|}{\|v_k + \eta - v_j\|}$$

$$\leq \frac{\|x - \frac{v_k+v_j}{2}\|\|\eta\| + \|\frac{\eta}{2}\|\|v_k - v_j\| + \|\frac{\eta}{2}\|\|\eta\|}{\|v_k + \eta - v_j\|}$$

$$\leq \frac{\|x - \frac{v_k+v_j}{2}\| + \frac{1}{2}\|v_k - v_j\| + \|\frac{\eta}{2}\|}{\|v_k - v_j\| - \|\eta\|}\|\eta\|$$

since $\|\eta\| \leq \frac{1}{2}\|\triangle S\|_{\min}$

$$\leq \frac{\|x - \frac{v_k+v_j}{2}\| + \frac{1}{2}\|v_k - v_j\| + \|v_k - v_j\|/4}{\|v_k - v_j\|/2}\|\eta\|$$

since $|X| < \infty$

$$\leq \frac{|X| + \frac{1}{2}\|v_k - v_j\| + \|v_k - v_j\|/4}{\|v_k - v_j\|/2}\|\eta\|$$
$$\leq \frac{|X| + \frac{1}{2}\|\triangle S\|_{\max} + \|\triangle S\|_{\max}/4}{\|\triangle S\|_{\min}/2}\|\eta\|$$
$$\leq \frac{2|X| + 2\|\triangle S\|_{\max}}{\|\triangle S\|_{\min}}\|\eta\|.$$

So Lipschitz constant

$$L = \frac{2|X| + 2\|\triangle S\|_{\max}}{\|\triangle S\|_{\min}}.$$

□

Proof of Theorem 18. Part II. τ is Lip. continuous in perturbation of vertices of S as functions in $\|\cdot\|_{L^p}$. Let $S^{(\eta)}$ have a perturbation of v_{k_0} for some k_0, $v_{k_0} + \eta$.

$$V_k^\eta = \{x \in X : d(x, v_k + \eta) < d(x, v_j)\ \forall j : v_k \neq v_j\}.$$

And

$$V_\epsilon^\eta = \{\sigma \in complex\,(\cup_k\ cl(V_k^\eta) - V_k^\eta) : d(\sigma, S) > \epsilon\}.$$

Then

$$\tau_{S^{(\eta)}}(\epsilon) = \min\left(\inf_{x\in S^{(\eta)}}\inf_{y\in V_\epsilon^\eta} d(x,y),\ \sup_{u,w\in X} d(u,w)\right)$$
$$\leq \min\left(\inf_{x\in S^{(\eta)}}\inf_{y\in V_\epsilon^0} d(x,y) + L\|\eta\|,\ \sup_{u,w\in X} d(u,w)\right)$$
$$\leq \min\left(\inf_{x\in S^{(\eta)}}\inf_{y\in V_\epsilon^0} d(x,y),\ \sup_{u,w\in X} d(u,w)\right) + L\|\eta\|$$
$$\leq \tau_S(\epsilon) + L\|\eta\|.$$

So

$$|\tau_{S^{(\eta)}}(\epsilon) - \tau_S(\epsilon)| \leq L\|\eta\|.$$

Now $\tau_{S^{(\eta)}}(\epsilon) = \tau_S(\epsilon)$ for $\epsilon \geq |X|$. Then

$$\begin{aligned}\|\tau_{S^{(\eta)}}(\cdot) - \tau_S(\cdot)\|_{L^p} &= \left(\int_0^\infty |\tau_{S^{(\eta)}}(\epsilon) - \tau_S(\epsilon)|^p\right)^{1/p} \\ &= \left(\int_0^{|X|} |\tau_{S^{(\eta)}}(\epsilon) - \tau_S(\epsilon)|^p\right)^{1/p} \\ &\leq L|X|^{1/p}\|\eta\|\end{aligned}$$

and

$$\sup_\epsilon |\tau_{S^{(\eta)}}(\epsilon) - \tau_S(\epsilon)| \leq L\|\eta\|.$$

□

We have showed the c-reach well-defined and robust improvement over the standard reach. However, so far it is defined on simplicial complexes, which are not as commonly studied as manifolds. Simplicial complexes are usually just an approximation to the object of interest which may be a manifold or stratification. Since simplicial complexes can approximate manifolds and stratifications, we will define c-reach with the limit of simplicial approximation, if it exists.

Definition 7.1.2. The c-reach, denoted $\tau_M(\cdot)$, on manifold or stratification $M \subset \mathbb{R}^m$ is given by

$$\tau_M(\epsilon) := \lim_{N\to\infty} \tau_{S_N}(\epsilon)$$

if the limit exists and where $S_N \subset M$, S_N finite, and $\lim_{N\to\infty} d_{HD}(S_N, M) = 0$.

Remark. *When stratifications of constant dimension are sampled statistically from a supported distribution, then with high probability, the above HD distance converges to 0 and the above definition applies. This is since the probability of* not *sampling an ϵ-ball is eventually arbitrarily small.*

The examples above indicate the c-reach and reach agree on manifolds which we will show. Consider the case where points $x, y \in M$ achieve the reach to z such that $d(x,z) = d(y,z)$. For any $\epsilon > 0$, for N large enough, there exists $\hat{x}, \hat{y} \in S_N$ with $d(x,\hat{x}) < \epsilon$ and $d(y,\hat{y}) < \epsilon$. Then there exists $\hat{z}$ such that $d(z,\hat{z}) < C\epsilon$ and $d(\hat{x},\hat{z}) = d(\hat{y},\hat{z})$ for some constant C. If there does not exist such an $x, y \in M$ then there exists $\tilde{x}, \tilde{y} \in M$ that are ϵ close to achieving the reach. And then the same argument holds. So under certain conditions on δ and N, $|\tau_M - \tau_{S_N}(\delta)| < \epsilon$. We can identify these conditions as follows. $\delta < \tau_M - C\epsilon - \epsilon$ is necessary otherwise the $\hat{z}$ vertex is filtered out. It is also necessary for $d_{HD}(M, S_N) < \delta$ so that the discretization is not mistaken for reach. For N large enough, there is a δ interval that approximates the reach, $d_{HD}(M, S_N) < \delta < \tau_M - C\epsilon - \epsilon$. Since for every $\epsilon : \tau_M/(1+C) > \epsilon > 0$, $\lim_{N\to\infty} |\tau_M - \tau_{S_N}(\delta)| < \epsilon$, there is equality $\tau_M = \tau_M(0^+)$. We have proved the follow theorem!

Theorem 19. *For a manifold or stratification M where $\tau_M(\cdot)$ is well-defined, $\tau_M(0^+) = \tau_M$.*

We see an example of exact convergence in the circle in Fig. 7.7 where the c-reach equals the reach (the radius) on the second step. To see an example of near convergence, let's take a look at the ellipse in Fig. 7.8. We see that due to a lack of samples, the c-reach is slightly higher than the reach for δ larger than the discretization artifacts. But as samples are taken near the endpoints of the ellipse, the c-reach recovers the reach.

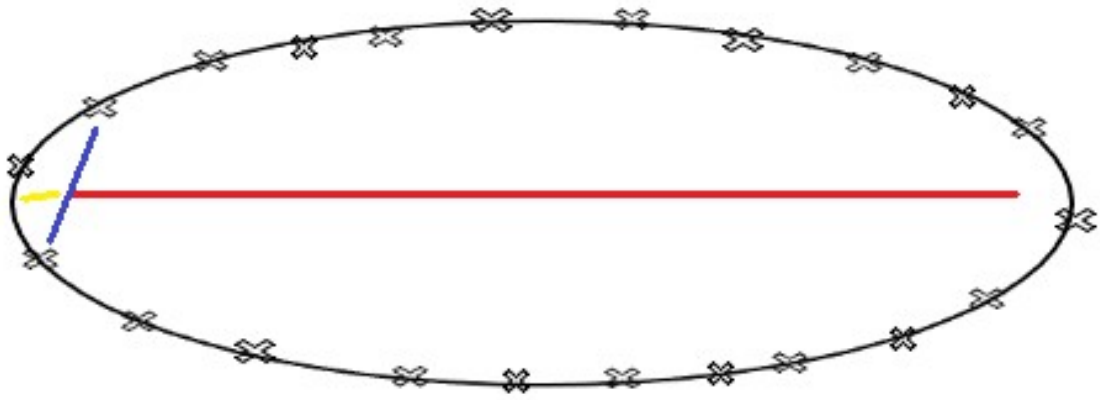


Figure 7.8: Manifold is ellipse in black. Medial axis is in red. Reach is achieved on endpoint of ellipse. Yellow line segment indicates reach. The samples are marked X. The blue line segment indicates c-reach which is half of blue line segment. We see how c-reach and reach converge with samples.

7.1.1 Noise

With noisy samples from a manifold, there is a trade-off in estimation error of the c-reach based on the noise covariance matrix and the curvature of the manifold. Samples with noise in a direction tangent to the manifold corresponds to noiseless samples of a manifold with approximately the same c-reach. Whereas samples with noise in a normal direction are equivalent to noiseless samples from a highly curved manifold. Here the difference in dimension between the manifold and the ambient space becomes very relevant. The larger the difference in dimension, the more likely the noise leaves the tangent plane. The more curved a manifold becomes, the lower the reach. Where the reach agrees with the c-reach, the c-reach becomes lower. Indeed, the c-reach at all levels generally decrease because the symmetry that gives rise to the medial axis is broken. We plot an example of sampling an ellipse with and without noise in Fig. 7.9.

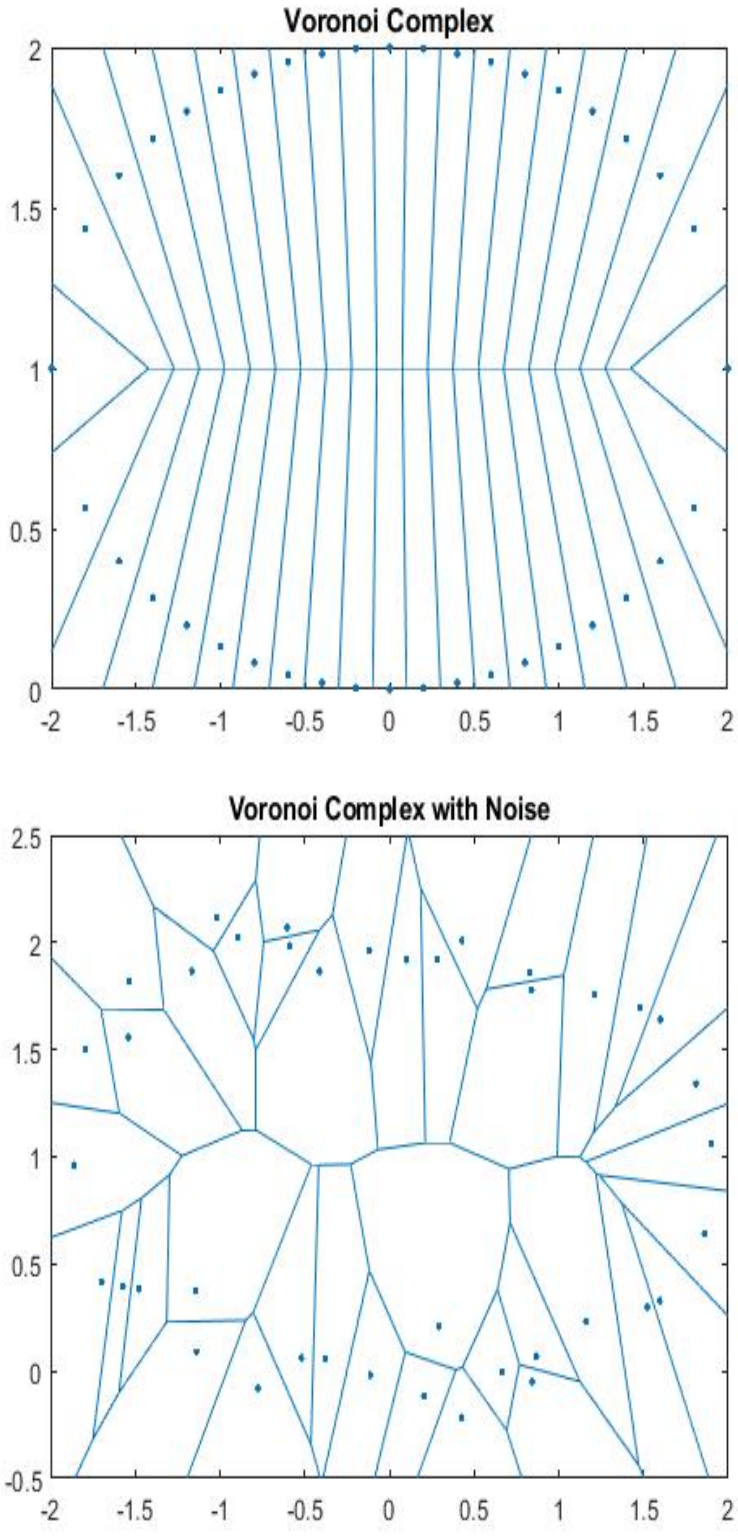


Figure 7.9: Top: Noiseless samples from ellipse are dotted in blue. Bottom: Samples with Gaussian noise (with identity covariance) are dotted in blue. Top and Bottom: Voronoi complexes are shown as blue graphs.

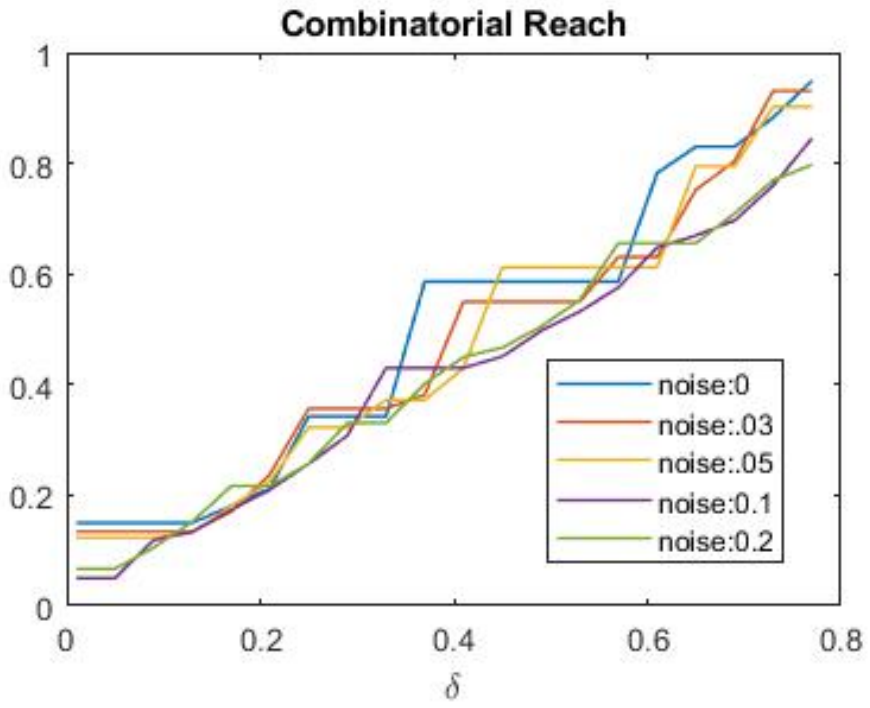


Figure 7.10: Plot shows c-reach of ellipse samples with various levels (coeff.) of additive Gaussian noise with identity covariance.

Noisy samples corrupt curvature information. C-reach estimates curvature of the manifold, but the more noise, the worse the estimate. In the limit of samples that are pure noise (and no signal), the c-reach shows no structure whatsoever. In Fig. 7.10, we plot the c-reach noisy samples of the ellipse from Fig. 7.9. We see that even small amounts of noise flatten the c-reach. Now we shall compute and plot the relative error as we stretch the ellipse from circle to straight line in Fig. 7.11. The relative error increases the flatter the ellipse becomes. This shows how stability increases with curvature.

The c-reach estimates give information about curvature and reach of manifolds which is very useful for applications. Curvature information allows bounding approximation error in manifold reconstruction. To do this, piece-wise linear interpolation is computed on pieces of

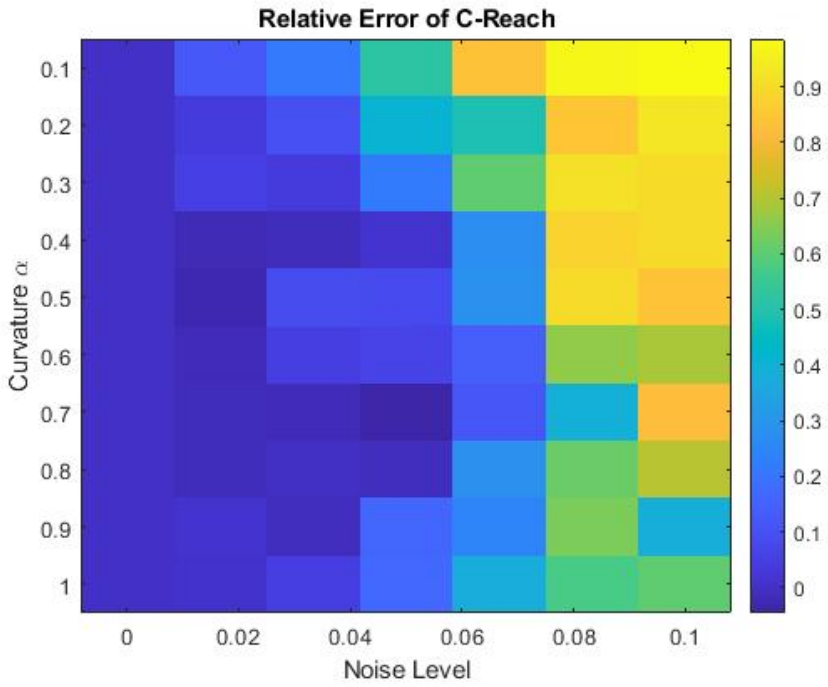


Figure 7.11: C-Reach discrete approximation error is plotted for different noise levels and ellipse curvatures α. The noisily samples are an ellipse segment parameterized by α where [0,1]=[line,circle].

the manifold. Refining the pieces small enough to limit the curvature error can be done once curvature is estimated. Another application for c-reach estimates is topology reconstruction. In topology reconstruction, the topology from the sampling manifold is estimated. A common method is persistent homology, see Chapter 6. In persistent homology, the approximation error can be bounded with the reach and other parameters [30]. The c-reach can be utilized here to improve or realize approximation bounds.

7.1.2 Open Problems

- Can Voronoi diagram be calculated in linear time in high dimensions?

- Can this calculation be generalized to approximate the mu-reach and improve persistent homology bounds?

Chapter 8

Hypergraph Reconstruction with Neural Connectome Applications

Graphs are often useful for representing systems and signals. With a graph representation, the optimal path between nodes can be calculated, for example with the A^* algorithm or Dijkstra's algorithm [22, 36]. Another use of graphs may be to predict the spread of information or pathogens through a network. Graphs can represent any elements and pairwise interactions. However, in some cases there are 3-element interactions or 4-element interactions. A graph can be generalized and given multi-edges that 'connect' more than 2 nodes in the graph. We call this a hypergraph which contains nodes and hyperedges [80]. For example, consider representing medical drugs as a graph. Let each drug be a node and let there be an edge when a pair of drugs can be taken together. This graph is lacking in that it cannot tell whether 3 drugs should be taken together. When we upgrade this graph to a hypergraph that has a hyperedge for every viable combination, all of the information can be represented and analyzed. In this chapter, we discuss *reconstructing* hypergraphs.

Then once a hypergraph can be reconstructed from statistical measurements, we can use its geometry and topology for analyzing datasets. The geometry and topology of a hypergraph is often so complex that we must use machine learning, see Chapter 2, to model datasets, as we do below.

Organisms with a nervous system have many nerve cells or neurons that receive and send electrical impulses. Neurons throughout the organism are connected to each other via synapses. These electrical impulses are signals that coordinate and trigger movement in a organism. The human brain contains an estimated 86 billion neurons [50]. Since neurons are connected to each other, we can model the neurons and their connections as a graph. Activity in certain regions of this graph are responsible for certain behaviors and capabilities. Studying the characteristics of this graph or even writing down this graph is an emerging area of study in computational neuroscience. Measuring which neurons are connected is also difficult without invasive techniques.

Functional magnetic resonance imaging (fMRI) since the 1990's has been used to investigate brain activity in a benign and noninvasive way [52]. fMRI detects oxygen usage by neurons with changing magnetic fields. fMRI is not sensitive enough to measure something as small as a neuron so a region of around 4mm by 4mm by 4mm is scanned. Typically, the subject's whole brain is scanned over these small regions which are called *voxels*. We can view the results as a 3D matrix, or array, of real numbers and each entry corresponds to a voxel or region in the subject. Each voxel represents an average of up to around a few million neurons and tens of billions of synapses. Implicit here is that the scan happens at some point in time. More accurately, the scan happens over around 1 second. This will miss events in the brain that happen in significantly less time than 1 second. Often this large

dataset is compressed further by grouping voxels into known regions of the brain [29].

A connectome is a graph describing the connections between neurons. By grouping neurons into voxels, a graph can be constructed from fMRI scan data. Each vertex of the graph is a voxel and each weighted edge is the correlation between the voxels. Correlation between voxels imply the neurons are directly or indirectly connected. This also assumes that neural activity is probabilistic. When the number of samples or measurements is low, the statistical confidence is low. To improve the confidence, voxels are often grouped into brain regions. Then each brain region has many samples and the confidence of the correlation is much greater. Additionally, the graph where the vertices correspond to few brain regions is much smaller allowing for more computationally intensive graph analysis methods.

Brain function and connectivity is a pressing mystery in medicine related to many diseases. Neural connectomes have been studied as graphs with graph theory methods including topological methods. Work has started on hypergraph models and methods where the geometry and topology is significantly different. We define a hypergraph called the *hyper-connectome* with joint information entropy and total correlation [87]. Brain fMRI scan data can be viewed as a hypergraph by using high order statistical methods. Hypergraph analysis can then yield a deeper, more informative analysis on brain connectomes. We build a brain connectome hypergraph from fMRI scans to identify indicators that are impossible to retrieve from a standard brain connectome graph. Note that the hyper-connectome is *not* a simplicial complex and thus most topological data analysis (TDA) methods, such as persistent homology, will not work here [103].

There has been a large amount of activity around this area in the recent past. In the 2000s, Darling and Norris studied large random hypergraphs [26] which is exactly what a

hyper-connectome is. Then in 2016, Munsell, Zu, Giusti, et al. considered different types of hypergraphs related to connectomes and neural data and various diseases [39, 71, 122]. Sparse linear regression has been used to predict hyperedges in hypergraphs and classify disease [44, 55, 63]. In 2018, hypergraph methods were compared and contrasted to TDA, sheaf methods, point cloud methods, and others by Purvine et al. [80]. Later, Sizemore et al. created a structural hypergraph of a mouse connectome [101] and analyzed the topology. On the other hand, others learned, or optimized, a hypergraph as opposed to direct calculation [115, 123]. Banka et al. learned autoencoder embeddings by hypergraphs [10]. More broadly, Aksoy et al. performed a general comparison between graphs and hypergraphs and their methods [5]. More recently, Stolz et al. analyzed the topology of connectomes in schizophrenic subjects versus normal subjects (siblings and non-siblings) [103].

Given two random variables, the Pearson correlation coefficient is

$$corr(X_1, X_2) := \frac{\mathbb{E}[(X_1 - \mathbb{E}X_1)(X_2 - \mathbb{E}X_2)]}{\sqrt{\mathbb{E}((X_1 - \mathbb{E}X_1)^2)}\sqrt{\mathbb{E}((X_2 - \mathbb{E}X_2)^2)}}.$$

This can be approximated given samples from distributions. The sample Pearson correlation coefficient, with n samples, is

$$corr(\hat{X}_1, \hat{X}_2) := \frac{\sum_i \frac{1}{n}(\hat{X}_{1,i} - \frac{1}{n}\sum_k \hat{X}_{1,k})(\hat{X}_{2,i} - \frac{1}{n}\sum_k \hat{X}_{2,k})}{\sqrt{[\sum_i \frac{1}{n}(\hat{X}_{1,i} - \frac{1}{n}\sum_k \hat{X}_{1,k})][\sum_j \frac{1}{n}(\hat{X}_{2,j} - \frac{1}{n}\sum_k \hat{X}_{2,k})]}}.$$

We will use this on fMRI samples to identify structure between brain regions. However, there are other statistics that we can use. The information entropy of a discrete random variable is

$$H(X) = -\sum_{x \in \mathbb{R}} p(x) \log p(x)$$

where p is the probability that $X = x$. Note that we use the convention that $0 \cdot \log(0) = 0$. So again this can be approximated with samples. The sample information entropy, with n

samples, is

$$H(\hat{X}) = -\sum_{x\in\mathbb{R}} \frac{1}{n} \sum_i \mathbb{1}_{\hat{X}_i=x}(\hat{X}_i) \log \left[\frac{1}{n} \sum_i \mathbb{1}_{\hat{X}_i=x}(\hat{X}_i) \right].$$

Now for any collection of discrete random variables, we can define the joint information entropy as

$$H(X_1, ..., X_n) = -\sum_{x_1,...,x_n\in\mathbb{R}} p(x_1, ..., x_n) \log p(x_1, ..., x_n).$$

The joint information entropy can be very useful for approximating how independent brain regions are versus how closely they collaborate in a predictable way. This is captured more directly by comparing the sum of entropies with the joint entropy. This difference is the mutual information or total correlation

$$C(X_1,X_2, ..., X_n) := [\sum_i H(X_i)] - H(X_1, ..., X_n).$$

We use the following well known simplification.

Proposition 8.0.1.

$$C(X_1, X_2, ..., X_n) = \sum_{x_1,x_2,...,x_n\in\mathbb{R}} p(x_1, x_2, ..., x_n) \log \frac{p(x_1, x_2, ..., x_n)}{p(x_1)p(x_2)...p(x_n)}.$$

Proof of Proposition 8.0.1.

$$C(X_1, X_2, ..., X_n) = [\sum_i H(X_i)] - H(X_1, ..., X_n)$$

$$= [\sum_{x_1,...,x_n \in \mathbb{R}} p(x_1, ..., x_n) \log p(x_1, ..., x_n)] - [\sum_i \sum_{x_i} p(x_i) \log p(x_i)]$$

$$= [\sum_{x_1,...,x_n \in \mathbb{R}} p(x_1, ..., x_n) \log p(x_1, ..., x_n)] - [\sum_i \sum_{x_i} \sum_{x_1,...,x_{i-1},x_{i+1},...,x_n \in \mathbb{R}} p(x_1, ..., x_n) \log p(x_i)]$$

$$= [\sum_{x_1,...,x_n \in \mathbb{R}} p(x_1, ..., x_n) \log p(x_1, ..., x_n)] - [\sum_{x_1,...,x_n \in \mathbb{R}} p(x_1, ..., x_n) \sum_i \log p(x_i)]$$

$$= \sum_{x_1,...,x_n \in \mathbb{R}} [p(x_1, ..., x_n) \log p(x_1, ..., x_n) - p(x_1, ..., x_n) \sum_i \log p(x_i)]$$

$$= \sum_{x_1,...,x_n \in \mathbb{R}} p(x_1, ..., x_n) \log \frac{p(x_1, ..., x_n)}{p(x_1)...p(x_n)}$$

□

Next we show how discrete random variables can approximate absolutely continuous random variables.

Theorem 20. *[87] Let Y_i be absolutely continuous random variables, $Y_i : \Omega \to \mathbb{R}$ measurable. The joint density p_Y can be approximated by simple functions arbitrarily close in integration. We use an approximation and we say $p_{\tilde{Y}_i}$ is a simple function. Set discrete random variables X_i to have a dirac for each term in simple function $p_{\tilde{Y}}$ with the coefficient so that the measures are the same:*

$$p_{\tilde{Y}} = \sum_k \alpha_k \mathbb{1}_{\Pi_i(a_{i,k}, b_{i,k})}, \quad p_X = \sum_k \beta_k \delta_{\Pi_i(b_{i,k} - a_{i,k})/2}, \quad \beta_k = \alpha_k \mathbb{P}_{\tilde{Y}}(\Pi_i(a_{i,k}, b_{i,k}))$$

For any $\epsilon > 0$, there is an discrete approximation with the total correlation

$$|C(Y_1, Y_2, ..., Y_n) - C(X_1, X_2, ..., X_n)| < \epsilon.$$

Proof of Theorem 20. Recall

$$p_{\tilde{Y}} = \sum_k \alpha_k \mathbb{1}_{\Pi_i(a_{i,k},b_{i,k})}, \quad p_X = \sum_k \beta_k \delta_{\Pi_i(b_{i,k}-a_{i,k})/2}, \quad \beta_k = \alpha_k \mathbb{P}_{\tilde{Y}}(\Pi_i(a_{i,k}, b_{i,k})).$$

The total correlation

$$C(\tilde{Y}_1, \tilde{Y}_2, ..., \tilde{Y}_n) = \int_{y_1,y_2,...,y_n \in \mathbb{R}} p_{\tilde{Y}}(y_1, y_2, ..., y_n) \log \frac{p_{\tilde{Y}}(y_1, y_2, ..., y_n)}{p_{\tilde{Y}}(y_1)p_{\tilde{Y}}(y_2)...p_{\tilde{Y}}(y_n)} dy_1...dy_n.$$

Write compactly as the integral of the simple function $\Phi_{\tilde{Y}}(y) = \sum_{k\in\mathbb{N}} \alpha_k \mathbb{1}_{\Pi_i(a_{i,k},b_{i,k})}(y)$

$$\begin{aligned}
C(\tilde{Y}_1, \tilde{Y}_2, ..., \tilde{Y}_n) &= \int_{y\in\mathbb{R}^n} \Phi_{\tilde{Y}}(y)dy = \int_{y\in\mathbb{R}^n} \sum_{k\in\mathbb{N}} \alpha_k \mathbb{1}_{\Pi_i(a_{i,k},b_{i,k})}(y)dy \\
&= \sum_{k\in\mathbb{N}} \alpha_k \int_{\Pi_i(a_{i,k},b_{i,k})} dy = \sum_{k\in\mathbb{N}} \beta_k \int \delta_{\Pi_i(b_{i,k}-a_{i,k})/2} \\
&= \sum_{x\in\mathbb{R}^n} \Phi_X(x) = \sum_{x_1,x_2,...,x_n\in\mathbb{R}} p_X(x_1, x_2, ..., x_n) \log \frac{p_X(x_1, x_2, ..., x_n)}{p_X(x_1)p_X(x_2)...p_X(x_n)} \\
&= C(X_1, X_2, ..., X_n).
\end{aligned}$$

□

With the justification of Theorem 20, we will approximate the total correlation from finite samples with Algorithm 12.

8.1 Simulation

First we will consider a small example where we know the distributions. We will setup the distributions where doing classification by hypergraph is in theory possible, which we

prove. For random variables X and Y, let

$$X_i \sim Bernoulli(\{-1, 1\}, 1/2), 1 \leq i \leq 3$$

and

$$Y = [Y_1, Y_2, Y_3] = [X_1X_2, X_2X_3, X_3X_1].$$

Then $\sigma_{Y_i}^2 = \mathbb{E}Y_i^2 = 1$. For $i \neq j$, let k, u, and v be such that $Y_i = X_kX_u$ and $Y_j = X_kX_v$. Then

$$corr(Y_i, Y_j) = \frac{\mathbb{E}[Y_iY_j]}{\sigma_{Y_i}\sigma_{Y_j}} = 0. \tag{8.1}$$

Proof of Equation (8.1).

$$corr(Y_i, Y_j) = \frac{\mathbb{E}[Y_iY_j]}{\sigma_{Y_i}\sigma_{Y_j}}$$

$$= \mathbb{P}(Y_i = 1, Y_j = 1) - \mathbb{P}(Y_i = -1, Y_j = 1) - \mathbb{P}(Y_i = 1, Y_j = -1) + \mathbb{P}(Y_i = -1, Y_j = -1)$$

$$= \mathbb{P}(X_1 = 1, X_2 = 1, X_3 = 1) + \mathbb{P}(X_1 = -1, X_2 = -1, X_3 = -1)$$

$$- 2\mathbb{P}(Y_i = -1, Y_j = 1) + \mathbb{P}(X_k = -1, X_u = 1, X_v = 1) + \mathbb{P}(X_k = 1, X_u = -1, X_v = -1)$$

$$= 1/8 + 1/8 - 2\mathbb{P}(X_k = -1, X_u = 1, X_v = -1) - 2\mathbb{P}(X_k = 1, X_u = -1, X_v = 1) + 1/8 + 1/8$$

$$= 0$$

□

This implies that the connectome Pearson correlation graph will not distinguish subject X from subject Y. However, the *total correlation* of Y is not 0.

Proposition 8.1.1. [87]

$$C(Y_1, Y_2, Y_3) = [\sum_i H(Y_i)] - H(Y_1, Y_2, Y_3) < 0.$$

Proof of Proposition 8.1.1.

$$C(Y_1, Y_2, Y_3) = [\sum_i H(Y_i)] - H(Y_1, Y_2, Y_3)$$

$$= [\sum_i \mathbb{P}(Y_i = 1) \log \mathbb{P}(Y_i = 1) + \mathbb{P}(Y_i = -1) \log \mathbb{P}(Y_i = -1)]$$

$$+ \mathbb{P}(Y_1 = 1, Y_2 = 1, Y_3 = 1) \log \mathbb{P}(Y_1 = 1, Y_2 = 1, Y_3 = 1)$$

$$+ \mathbb{P}(Y_1 = 1, Y_2 = 1, Y_3 = -1) \log \mathbb{P}(Y_1 = 1, Y_2 = 1, Y_3 = -1)$$

$$+ \mathbb{P}(Y_1 = 1, Y_2 = -1, Y_3 = 1) \log \mathbb{P}(Y_1 = 1, Y_2 = -1, Y_3 = 1)$$

$$+ \mathbb{P}(Y_1 = 1, Y_2 = -1, Y_3 = -1) \log \mathbb{P}(Y_1 = 1, Y_2 = -1, Y_3 = -1)$$

$$+ \mathbb{P}(Y_1 = -1, Y_2 = 1, Y_3 = 1) \log \mathbb{P}(Y_1 = -1, Y_2 = 1, Y_3 = 1)$$

$$+ \mathbb{P}(Y_1 = -1, Y_2 = 1, Y_3 = -1) \log \mathbb{P}(Y_1 = -1, Y_2 = 1, Y_3 = -1)$$

$$+ \mathbb{P}(Y_1 = -1, Y_2 = -1, Y_3 = 1) \log \mathbb{P}(Y_1 = -1, Y_2 = -1, Y_3 = 1)$$

$$+ \mathbb{P}(Y_1 = -1, Y_2 = -1, Y_3 = -1) \log \mathbb{P}(Y_1 = -1, Y_2 = -1, Y_3 = -1)$$

$$= [\sum_i 1/2 \log 1/2 + 1/2 \log 1/2]$$

$$+ \mathbb{P}(Y_1 = 1, Y_2 = 1, Y_3 = 1) \log \mathbb{P}(Y_1 = 1, Y_2 = 1, Y_3 = 1)$$

$$+ \mathbb{P}(Y_1 = 1, Y_2 = -1, Y_3 = -1) \log \mathbb{P}(Y_1 = 1, Y_2 = -1, Y_3 = -1)$$

$$+ \mathbb{P}(Y_1 = -1, Y_2 = 1, Y_3 = -1) \log \mathbb{P}(Y_1 = -1, Y_2 = 1, Y_3 = -1)$$

$$+ \mathbb{P}(Y_1 = -1, Y_2 = -1, Y_3 = 1) \log \mathbb{P}(Y_1 = -1, Y_2 = -1, Y_3 = 1)$$

$$= [3 \log 1/2] + 1/4 \log 1/4 + 1/4 \log 1/4 + 1/4 \log 1/4 + 1/4 \log 1/4$$

$$= 3 \log 1/2 + \log 1/4 < 0.$$

□

In this case, a discriminator can distinguish subject Y from subject X with total correlation but would fail to distinguish when using Pearson correlations. If this was a neural connectome, only the hyper-connectome could classify the subjects. We calculate the hypergraph with Algorithm 12. Then we classify the data with a linear support vector machine [107]. We give the results in Table 8.1. We find that the graph does not contain the information needed to classify the subject as a member of X versus Y. We find that the hypergraph does contain the information and the classifier successfully distinguishes X from Y.

Table 8.1: Linear support vector machine classification of X versus Y from above. Hypergraph threshold $\epsilon = 10^{-5}$, dimension $d = 3$, and variable count 3. Subjects are 1000 subject from X and 1000 subjects from Y. There are 20 samples of each subject. The training/testing split is random 50%.

	Training Accuracy	**Testing Accuracy**	**F1 Score**
Graph	51%	49%	0.66
Hypergraph	100%	100%	1

8.2 Schizophrenia Data

We next compute the hyper-connectome on real data. We utilize a schizophrenia (schiz.) fMRI dataset, see [114], consisting of 104 patients with schizophrenia and 124 healthy, normal controls. Between the groups, the age and gender differences are minimal (schiz.: age 36.88 $\pm$ 14.17 with 62 males, 41 females, 1 other, and Normal: age 33.75 $\pm$ 14.22 with 61 males and

63 females). The fMRI aquisition details and preprocessing details are laid out in Adhikari et al. [3]. This dataset consists of activity in 246 regions of interest (ROI) which we call the *ROI variables* [29]. We create the connectome and hyper-connectome with a subset of these ROI variables.

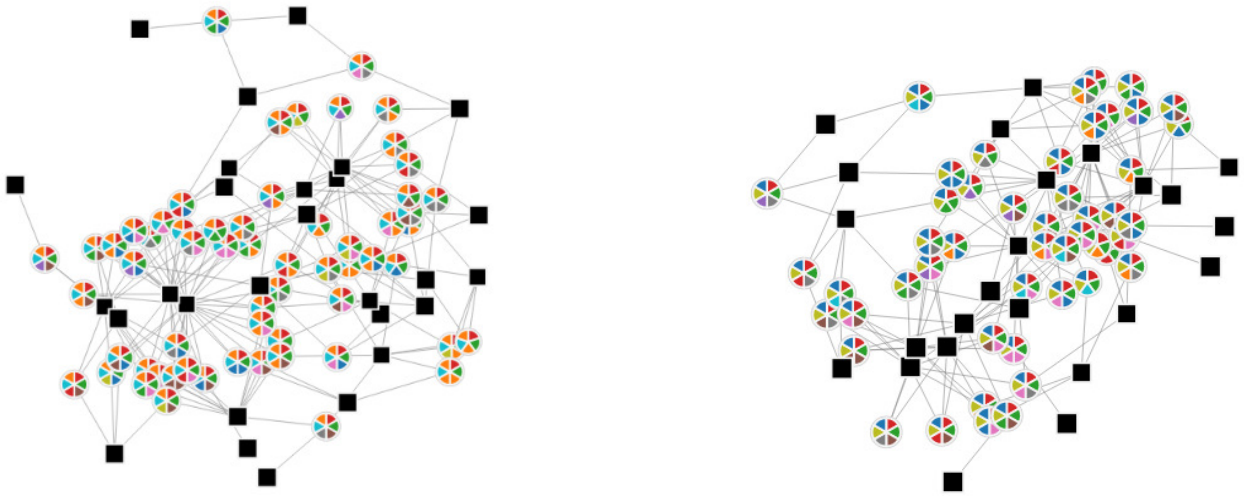
(a) Normal Subject Connectome

(b) Schiz. Subject Connectome

Figure 8.1: Connectome graph of normal vs. schiz. subject with 30 ROI nodes and correlation weighted edges.

We visualize connectomes by plotting the graph for a normal subject and schiz. subject in Fig. 8.1a and Fig. 8.1b. In this graph, the nodes are the brain regions and the edges are the absolute value of Pearson correlation between regions, where length and width indicate the weight of the edge. In Fig. 8.2a and Fig. 8.2b, we visualize the hyper-connectomes of the same two subjects. In this graph, the square nodes are the brain regions and the circles are significant hyperedges between multiple nodes [119]. We see that the connectomes are highly clustered with few outliers versus the hyper-connectomes which are have nodes covering the connectedness spectrum.

In Fig. 8.3 and Fig. 8.4, we show the corresponding adjacency matrices to the graphs in



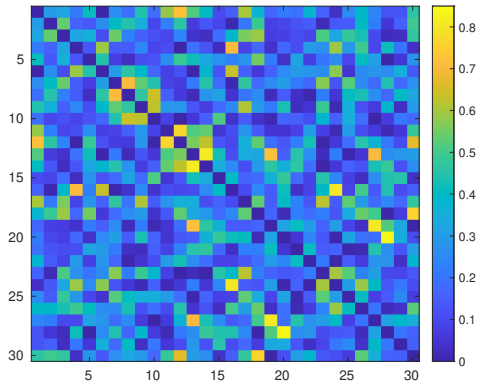
(a) Normal Subject Connectome

(b) Schiz. Subject Connectome

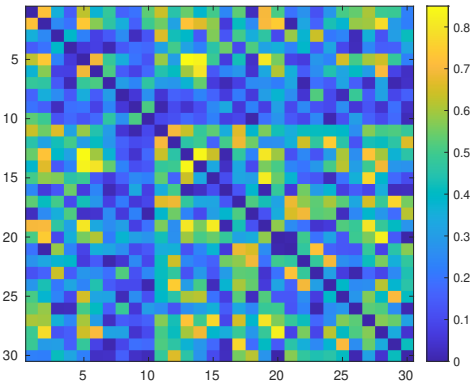
Figure 8.2: (a) and (b): Hyper-Connectome of normal vs. schiz. subject, plotting significant (weight $> 2^8$) edges (circles) for 30 ROI nodes (squares).

Fig. 8.1 and Fig. 8.2. For each pair of ROI, we sum the weights of all common hyperedges to produce the adjacency matrices in Fig. 8.4a and Fig. 8.4b. We see that the magnitudes of Fig. 8.3a and Fig. 8.3b are similar (in 0 to 1) while the maximums of Fig. 8.4a and Fig. 8.4b differ significantly (5000 vs. 5500). We calculated the hypergraph with Algorithm 12. The hypergraph threshold $\epsilon = 10^{-5}$, dimension $d = 3$, and ROI variables are 30 brain regions. Samples used are the first 20 in the time series.

We showed above that the connectome and hyper-connectome are differ greatly in information content. The next question is how useful this is for distinguishing normal subjects and schizophrenic subjects. We vectorize the upper triangle of the connectome adjacency matrix and train a linear support vector machine to classify the subjects [107]. After training, we calculate the test accuracy and F1 score on unseen data. We follow the same procedure with the hyper-connectome. We report the results in Table 8.2. We find an increase in the testing



(a) Normal Subject Pearson Correlation Matrix



(b) Schiz. Subject Pearson Correlation Matrix

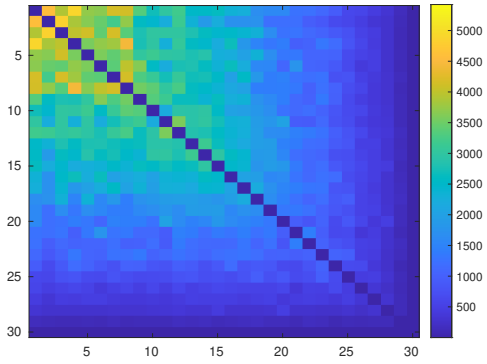
Figure 8.3: Pearson correlation matrix of normal vs. schiz. subject.

accuracy and in the F1 score from using the hyper-connectome versus the connectome.

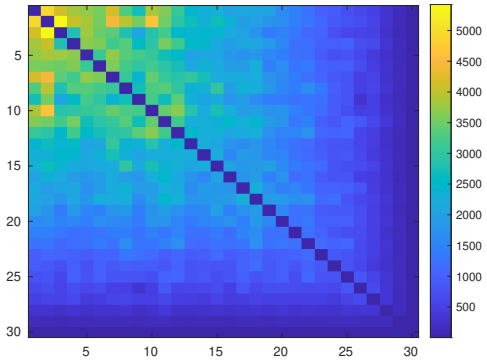
Table 8.2: Linear support vector machine prediction of schiz. Hypergraph threshold $\epsilon = 10^{-5}$, dimension $d = 3$, and 61 ROI variables considered. Samples over time are 30. The training/testing split is random 50%. Calculations are average of 10 independent trials.

	Training Accuracy	**Testing Accuracy**	**F1 Score**
Graph	100%	51%	0.44
Hypergraph	100%	57%	0.52

We have introduced the entropic hyper-connectome as a useful concept to study neuronal structure, function, and abnormalities. We have demonstrated this with fMRI data in vivo. From theory, we have defined the hypergraph and proved that it can be necessary to detect various mixture distributions. We visualized the connectome and hyper-connectome and



(a) Normal Subject Total Correlation Tensor



(b) Schiz. Subject Total Correlation Tensor

Figure 8.4: Total correlation tensor of normal vs. schiz. subject projected via summation to matrix.

see significant differences. Finally, we trained a classifier to show that the hyperedges can improve classification.

8.3 Open Problems

- Can a sparse hypergraph be calculated without checking each node tuple?
- Can Bayesian iterations provably approximate total correlation while decreasing computation?

Algorithm 12: Total Correlation

Input:

$M \in \mathbb{N}$: variables

$N \in \mathbb{N}$: samples

$d \in \mathbb{N}$: dimensions

$X \in \mathbb{R}^{M \times N}$: measurements

$\epsilon \in \mathbb{R}$: threshold

Output:

$T \in \mathbb{R}^{M^d}$: total correlation

Begin:

$T = 0 \in \mathbb{R}^{M^d}$

for $1 \leq i_1 \leq M$ **do**

  for $i_1 \leq i_2 \leq M$ **do**

    ...

      for $i_{d-1} \leq i_d \leq M$ **do**

        for $1 \leq s_1 \leq N$ **do**

          $\omega_1 = X(i_1, s_1)$

          $p_1(\omega_1) = \frac{1}{N} \sum_j \mathbb{1}_{|X(i_1,j)-\omega_1|<\epsilon}(X)$

          for $1 \leq s_2 \leq N$ **do**

            $\omega_2 = X(i_2, s_2)$

            $p_2(\omega_2) = \frac{1}{N} \sum_j \mathbb{1}_{|X(i_2,j)-\omega_2|<\epsilon}(X)$

            ...

            for $1 \leq s_d \leq N$ **do**

              $\omega_d = X(i_d, s_d)$

              $p_d(\omega_d) = \frac{1}{N} \sum_j \mathbb{1}_{|X(i_d,j)=\omega_d|<\epsilon}(X)$

              $p(\omega_1, \omega_2, ..., \omega_d) = \frac{1}{N} \sum_j \mathbb{1}_{|X(i_1,j)-\omega_1|<\epsilon,...,|X(i_d,j)-\omega_d|<\epsilon}(X)$

              $T(i_1, i_2, ..., i_d) =$

               $T(i_1, i_2, ..., i_d) + p(\omega_1, \omega_2, ..., \omega_d) \log \frac{p(\omega_1,\omega_2,...,\omega_d)}{p_1(\omega_1)p_2(\omega_2)...p_d(\omega_d)}$

            end

            ...

          end

        end

      ...

  end

end

Bibliography

[1] Eddie Aamari, Jisu Kim, Frédéric Chazal, Bertrand Michel, Alessandro Rinaldo, and Larry Wasserman. Estimating the reach of a manifold. *Electronic journal of statistics*, 13(1):1359–1399, 2019.

[2] Angela Adamo, JE Ryon, Matteo Messa, Hwihyun Kim, Kathryn Grasha, DO Cook, Daniela Calzetti, Janice C Lee, BC Whitmore, BG Elmegreen, and others. Legacy ExtraGalactic UV survey with the hubble space telescope: Stellar cluster catalogs and first insights into cluster formation and evolution in NGC 628. *The Astrophysical Journal*, 841(2):131, 2017. doi: 10.3847/1538-4357/aa7132.

[3] Bhim M. Adhikari, L. Elliot Hong, Hemalatha Sampath, Joshua Chiappelli, Neda Jahanshad, Paul M. Thompson, Laura M. Rowland, Vince D. Calhoun, Xiaoming Du, Shuo Chen, and Peter Kochunov. Functional network connectivity impairments and core cognitive deficits in schizophrenia. *Human Brain Mapping*, 40(16):4593–4605, 2019. doi: 10.1002/hbm.24723.

[4] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. *JMLR Workshop and Conference Proceedings*, page 26, 2012.

[5] Sinan G. Aksoy, Cliff Joslyn, Carlos Ortiz Marrero, Brenda Praggastis, and Emilie Purvine. Hypernetwork science via high-order hypergraph walks. *EPJ Data Sci.*, 9(1): 16, 2020. doi: 10.1140/epjds/s13688-020-00231-0.

[6] Paolo Aluffi. *Algebra: chapter 0*, volume 104. American Mathematical Soc., 2009.

[7] Susan Athey and Stefan Wager. Policy learning with observational data. *Econometrica*, 89(1):133–161, 2021.

[8] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, 47(2):235–256, 2002. ISSN 08856125. doi: 10.1023/A:1013689704352.

[9] Radu Balan, Kasso A. Okoudjou, Michael Rawson, Yang Wang, and Rui Zhang. *Optimal l1 Rank One Matrix Decomposition*, pages 21–41. Springer International Publishing, Cham, 2021. ISBN 978-3-030-61887-2. doi: 10.1007/978-3-030-61887-2.

[10] Alin Banka, Inis Buzi, and Islem Rekik. Multi-view brain hyperconnectome autoencoder for brain state classification. In Islem Rekik, Ehsan Adeli, Sang Hyun Park,

and Maria del C. Valdés Hernández, editors, *Predictive Intelligence in Medicine*, pages 101–110, Cham, 2020. Springer International Publishing. ISBN 978-3-030-59354-4.

[11] John J Benedetto. *Harmonic analysis and applications*. CRC Press, 1997. ISBN 9781003068839.

[12] Norman L. Biggs. *Discrete mathematics*. Oxford University Press, 2002.

[13] Stephen P. Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004. ISBN 978-0-521-83378-3.

[14] D. Calzetti, J. C. Lee, E. Sabbi, A. Adamo, L. J. Smith, J. E. Andrews, L. Ubeda, S. N. Bright, D. Thilker, A. Aloisi, and et al. LEGACY EXTRAGALACTIC UV SURVEY (LEGUS) WITH THE HUBBLE SPACE TELESCOPE. i. SURVEY DESCRIPTION. *The Astronomical Journal*, 149(2):51, 2015. doi: 10.1088/0004-6256/149/2/51.

[15] Howard E Campbell. *The structure of arithmetic*. Appleton-Century-Crofts: New York, 1970.

[16] Emmanuel J. Candes and Michael B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008. doi: 10.1109/MSP.2007.914731.

[17] Gunnar Carlsson, Afra Zomorodian, Anne Collins, and Leonidas J. Guibas. PERSISTENCE BARCODES FOR SHAPES. *International Journal of Shape Modeling*, 11(2): 149–187, 2005. ISSN 0218-6543, 1793-639X. doi: 10.1142/S0218654305000761.

[18] Marine Carpuat and Dekai Wu. Improving statistical machine translation using word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.

[19] Frédéric Chazal. High-dimensional topological data analysis, 2016.

[20] Minshuo Chen, Hao Liu, Wenjing Liao, and Tuo Zhao. Doubly robust off-policy learning on low-dimensional manifolds by deep neural networks. *Submitted to Operations Research, under revision*, 2020.

[21] Harish Chintakunta, Michael Robinson, and Hamid Krim. Introduction to the special session on topological data analysis, icassp 2016. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6410–6414, 2016. doi: 10.1109/ICASSP.2016.7472911.

[22] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009. ISBN 9780262033848.

[23] T. J. Cornwell and K. F. Evans. A simple maximum entropy deconvolution algorithm. *Astronomy and Astrophysics*, 143:77–83, 1985.

[24] Tim J. Cornwell. Multiscale CLEAN deconvolution of radio synthesis images. *IEEE Journal of selected topics in signal processing*, 2(5):793–801, 2008.

[25] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26:2292–2300, 2013.

[26] R. W. R. Darling and J. R. Norris. Structure of large random hypergraphs. *The Annals of Applied Probability*, 15(1A):125 – 152, 2005. doi: 10.1214/105051604000000567.

[27] Herbert Edelsbrunner and Dmitriy Morozov. Persistent homology: theory and practice. Technical report, Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2012.

[28] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 454–463. IEEE, 2000.

[29] Lingzhong Fan, Hai Li, Junjie Zhuo, Yu Zhang, Jiaojian Wang, Liangfu Chen, Zhengyi Yang, Congying Chu, Sangma Xie, Angela R. Laird, Peter T. Fox, Simon B. Eickhoff, Chunshui Yu, and Tianzi Jiang. The human brainnetome atlas: A new brain atlas based on connectional architecture. *Cerebral Cortex*, 26(8):3508–3526, 07 2016. ISSN 1047-3211. doi: 10.1093/cercor/bhw157.

[30] Brittany Terese Fasy, Fabrizio Lecci, Alessandro Rinaldo, Larry Wasserman, Sivaraman Balakrishnan, and Aarti Singh. Confidence sets for persistence diagrams. *The Annals of Statistics*, 42(6):2301 – 2339, 2014. doi: 10.1214/14-AOS1252.

[31] Herbert Federer. Curvature measures. *Trans. Amer. Math. Soc.*, 93:418–491, 1959. doi: 10.1090/S0002-9947-1959-0110078-1.

[32] Abraham Adolf Fraenkel, Yehoshua Bar-Hillel, and Azriel Levy. *Foundations of set theory*. Elsevier, 1973.

[33] Jade Freeman and Michael Rawson. Top-K ranking deep contextual bandits for information selection systems. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2209–2214. IEEE, 2021.

[34] Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya-Polo, and Tomaso A Poggio. Learning with a wasserstein loss. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[35] Patrizio Frosini. A distance for similarity classes of submanifolds of a euclidean space. *Bulletin of the Australian Mathematical Society*, 42(3):407–415, 1990.

[36] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated planning and acting*. Cambridge University Press, 2016.

[37] Robert Ghrist. Barcodes: The persistent topology of data. *Bulletin of the American Mathematical Society*, 45(1):61–76, 2007. ISSN 0273-0979. doi: 10.1090/S0273-0979-07-01191-3.

[38] Robert Ghrist. Homological algebra and data. *Math. Data*, 25:273, 2018.

[39] Chad Giusti, Robert Ghrist, and Danielle S. Bassett. Two's company, three (or more) is a simplex. *Journal of Computational Neuroscience*, 41(1):1–14, Aug 2016. ISSN 1573-6873. doi: 10.1007/s10827-016-0608-6.

[40] K Grasha, D Calzetti, Angela Adamo, RC Kennicutt, BG Elmegreen, Matteo Messa, DA Dale, K Fedorenko, S Mahadevan, EK Grebel, and others. The spatial relation between young star clusters and molecular clouds in m51 with LEGUS. *Monthly Notices of the Royal Astronomical Society*, 483(4):4707–4723, 2019. doi: 10.1093/mnras/sty3424.

[41] Hermann Grassmann. *Lehrbuch der Arithmetik für höhere Lehranstalten*. Th. Chr. Fr. Enslin, 1861.

[42] Mark A Griswold, Peter M Jakob, Robin M Heidemann, Mathias Nittka, Vladimir Jellus, Jianmin Wang, Berthold Kiefer, and Axel Haase. Generalized autocalibrating partially parallel acquisitions (grappa). *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 47(6):1202–1210, 2002.

[43] Stephen F Gull and Geoff J Daniell. Image reconstruction from incomplete and noisy data. *Nature*, 272(5655):686–690, 1978.

[44] Hao Guo, Yao Li, Yong Xu, Yanyi Jin, Jie Xiang, and Junjie Chen. Resting-state brain functional hyper-network construction based on elastic net and group lasso methods. *Frontiers in Neuroinformatics*, 12, 2018. ISSN 1662-5196. doi: 10.3389/fninf.2018.00025.

[45] László Györfi, Michael Kohler, Adam Krzyżak, and Harro Walk. *A Distribution-Free Theory of Nonparametric Regression*. Springer Series in Statistics. Springer New York, 2002. ISBN 978-0-387-95441-7 978-0-387-22442-8. doi: 10.1007/b97848.

[46] Mark Harris. Inside pascal: Nvidia's newest computing platform. *NVIDIA Developer Blog*, 2016.

[47] A. Hatcher and Cambridge University Press. *Algebraic Topology*. Cambridge University Press, 2002. ISBN 9780521795401. URL `https://books.google.com/books?id=BjKs86kosqgC`.

[48] Jean-Claude Hausmann. On the vietoris-rips complexes and a cohomology theory for metric spaces. *Prospects in topology (Princeton, NJ, 1994) MR1368659*, pages 175–188, 1995.

[49] Heather Haynes and William Holmes. *The Emergence of Magnetic Resonance Imaging (MRI) for 3D Analysis of Sediment Beds*, chapter 1. Wiley & Sons, 2013. ISBN 2047-0371.

[50] Suzana Herculano-Houzel. The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost. *Proceedings of the National Academy of Sciences*, 109(supplement_1):10661–10668, 2012. doi: 10.1073/pnas.1201895109.

[51] Feng-Hsiung Hsu. *Behind Deep Blue: Building the computer that defeated the world chess champion*. Princeton University Press, 2002.

[52] Scott A Huettel, Allen W Song, Gregory McCarthy, et al. *Functional magnetic resonance imaging*, volume 1. Sinauer Associates Sunderland, MA, 2004.

[53] Jakob Hultgren. Lecture notes on Optimal Transport delivered at the University of Maryland at College Park, 2021.

[54] JA Högbom. Aperture synthesis with a non-regular distribution of interferometer baselines. *Astronomy and Astrophysics Supplement Series*, 15:417, 1974.

[55] Biao Jie, Chong-Yaw Wee, Dinggang Shen, and Daoqiang Zhang. Hyper-connectivity of functional networks for brain disease diagnosis. *Medical Image Analysis*, 32:84–100, 2016. ISSN 1361-8415. doi: 10.1016/j.media.2016.03.003.

[56] Leonid V Kantorovich. On the translocation of masses. In *Dokl. Akad. Nauk. USSR (NS)*, volume 37, pages 199–201, 1942.

[57] Yann LeCun. The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`, 2021.

[58] John M Lee. *Introduction to Smooth Manifolds*. Springer, New York, NY, 2013. ISBN 978-1-4419-9981-8. doi: 10.1007/978-1-4419-9982-5.

[59] Jan Lellmann, Dirk A. Lorenz, Carola Schönlieb, and Tuomo Valkonen. Imaging with kantorovich-rubinstein discrepancy. *SIAM Journal on Imaging Sciences*, 7(4):2833–2859, 2014. ISSN 1936-4954. doi: 10.1137/140975528.

[60] Michael Lesnick. Studying the shape of data using topology. *The Institute Letter*, pages 10–11, 2013.

[61] F. Li, T. J. Cornwell, and F. de Hoog. The application of compressive sampling to radio astronomy: I. deconvolution. *Astronomy & Astrophysics*, 528:A31, 2011. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201015045.

[62] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. *Proceedings of the 19th international conference on World wide web - WWW '10*, page 661, 2010. doi: 10.1145/1772690.1772758.

[63] Yang Li, Jingyu Liu, Xinqiang Gao, Biao Jie, Minjeong Kim, Pew-Thian Yap, Chong-Yaw Wee, and Dinggang Shen. Multimodal hyper-connectivity of functional networks using functionally-weighted lasso for mci classification. *Medical Image Analysis*, 52: 80–96, 2019. ISSN 1361-8415. doi: 10.1016/j.media.2018.11.006.

[64] Saunders Mac Lane. *Categories for the working mathematician*. Number 5 in Graduate texts in mathematics. Springer, 2nd ed edition, 1998. ISBN 978-0-387-98403-2.

[65] Julian Maclaren, Michael Herbst, Oliver Speck, and Maxim Zaitsev. Prospective motion correction in brain imaging: a review. *Magnetic resonance in medicine*, 69(3): 621–636, 2013.

[66] Paul Michel, Abhilasha Ravichander, and Shruti Rijhwani. Does the geometry of word embeddings help document classification? A case study on persistent homology-based representations. In *Proceedings of the 2nd Workshop on Representation Learning for NLP, Rep4NLP@ACL 2017*, pages 235–240. Association for Computational Linguistics, 2017. doi: 10.18653/v1/w17-2628.

[67] Kanika Mittal and Amita Jain. Word sense disambiguation method using semantic similarity measures and owa operator. *ICTACT Journal on Soft Computing*, 5(2), 2015.

[68] Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences de Paris*, 1781.

[69] Marston Morse. Rank and span in functional topology. *Annals of Mathematics*, pages 419–454, 1940.

[70] James R. Munkres. *Topology*. Pearson modern classic. Pearson, second edition edition, 2000. ISBN 978-0-13-468951-7.

[71] Brent C. Munsell, Guorong Wu, Yue Gao, Nicholas Desisto, and Martin Styner. Identifying relationships in functional and structural connectome data using a hypergraph learning method. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, pages 9–17, Cham, 2016. Springer International Pub. ISBN 978-3-319-46723-8.

[72] N. Anurag Murty, Vijay Natarajan, and Sathish Vadhiyar. Efficient homology computations on multicore and manycore systems. In *20th Annual International Conference on High Performance Computing*, pages 333–342, 2013. doi: 10.1109/HiPC.2013.6799139.

[73] Michael K Ng, Huanfeng Shen, Edmund Y Lam, and Liangpei Zhang. A total variation regularization based super-resolution reconstruction algorithm for digital video. *EURASIP Journal on Advances in Signal Processing*, 2007:1–16, 2007.

[74] Gregory Ongie, Ajil Jalal, Christopher A. Metzler, Richard G. Baraniuk, Alexandros G. Dimakis, and Rebecca Willett. Deep learning techniques for inverse problems in imaging. *IEEE Journal on Selected Areas in Information Theory*, 1(1):39–56, 2020. doi: 10.1109/JSAIT.2020.2991563.

[75] Giuseppe Peano. *Arithmetices principia: Nova methodo exposita*. Fratres Bocca, 1889.

[76] C. S. Peirce. On the logic of number. *American Journal of Mathematics*, 4(1):85–95, 1881. ISSN 00029327, 10806377. URL http://www.jstor.org/stable/2369151.

[77] Jose A Perea. A brief history of persistence. *Morfismos*, 23(1):1–16, 2019.

[78] Gabriel Peyré and Marco Cuturi. Computational optimal transport: With applications to data science. *Foundations and Trends in Machine Learning*, 11(5):355–607, 2019. ISSN 1935-8237. doi: 10.1561/2200000073.

[79] David M W Powers. Parallelized quicksort and radixsort with optimal speedup. In *Proceedings Of International Conference On Parallel Computing Technologies. Novosibirsk.*, pages 167–176. World Scientific, 1991.

[80] Emilie Purvine, Sinan Aksoy, Cliff Joslyn, Kathleen Nowak, Brenda Praggastis, and Michael Robinson. A topological approach to representational data models. In Sakae Yamamoto and Hirohiko Mori, editors, *Human Interface and the Management of Information. Interaction, Visualization, and Analytics*, pages 90–109, Cham, 2018. Springer International Publishing. ISBN 978-3-319-92043-6.

[81] Gustavo Pérez, Matteo Messa, Daniela Calzetti, Subhransu Maji, Dooseok E. Jung, Angela Adamo, and Mattia Sirressi. StarcNet: Machine learning for star cluster identification. *The Astrophysical Journal*, 907(2):100, 2021. ISSN 1538-4357. doi: 10.3847/1538-4357/abceba.

[82] Michael Rawson and Radu Balan. Convergence guarantees for deep epsilon greedy policy learning. *arXiv:2112.03376*, 2021.

[83] Michael Rawson and Jade Freeman. Deep upper confidence bound algorithm for contextual bandit ranking of information selection. *Proceedings of Joint Statistical Meetings (JSM), Statistical Learning and Data Science Section, 2021*, 2021. URL https://arxiv.org/abs/2110.04127.

[84] Michael Rawson and Jakob Hultgren. Optimal transport for super resolution applied to astronomy imaging. *Proceedings of EUSIPCO, 2022*, 2022. URL https://arxiv.org/abs/2202.05354. [Under Review].

[85] Michael Rawson, Samuel Dooley, Mithun Bharadwaj, and Rishabh Choudhary. Topological data analysis for word sense disambiguation. *arXiv:2203.00565*, 2022.

[86] Michael G. Rawson. Linear run time of persistent homology computation with GPU parallelization. *arXiv:2203.02527*, 2022.

[87] Michael G. Rawson. Entropic hyper-connectomes computation and analysis. *Proceedings of SIAM International Conference on Data Mining (SDM22)*, 2022. URL https://arxiv.org/abs/2203.00519.

[88] Michael G. Rawson and Michael Robinson. A combinatorial reach for resolving inadequacies of reach, 2022. [In Preparation].

[89] Michael G. Rawson, Xiaoke Wang, Radu Balan, and Thomas Ernst. MGRAPPA: Motion corrected GRAPPA for MRI. *Magnetic Resonance in Medicine*, 2022. [In Preparation].

[90] Dedekind Richard and Ewald William. Was sind und was sollen die zahlen. *Ewald*, 2: 787–833, 1888.

[91] Vanessa Robins. Towards computing homology from finite approximations. In *Topology proceedings*, volume 24, pages 503–532, 1999.

[92] Michael Robinson. *Topological Signal Processing*. Springer Berlin Heidelberg, 2014.

[93] Michael Robinson and Christopher Capraro. Super-resolving star clusters with sheaves. *arXiv:2106.08123*, 2021.

[94] Walter Rudin. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1964.

[95] Walter Rudin. *Real and complex analysis*. McGraw-Hill, 3rd ed edition, 1987. ISBN 978-0-07-054234-1.

[96] Walter Rudin. *Functional analysis*. International series in pure and applied mathematics. McGraw-Hill, 2nd ed edition, 1991. ISBN 978-0-07-054236-5.

[97] Ishrat Rahman Sami and Katayoun Farrahi. A simplified topological representation of text for local and global context. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1451–1456. ACM, 2017.

[98] Ketki Savle, Wlodek Zadrozny, and Minwoo Lee. Topological data analysis for discourse semantics? In *Proceedings of the 13th International Conference on Computational Semantics-Student Papers*, pages 34–43, 2019.

[99] Zuowei Shen, Haizhao Yang, and Shijun Zhang. Optimal approximation rate of ReLU networks in terms of width and depth. *Journal de Mathématiques Pures et Appliquées*, 157:101–135, 2022. ISSN 0021-7824. doi: 10.1016/j.matpur.2021.07.009.

[100] Ravi Sinha and Rada Mihalcea. Unsupervised graph-basedword sense disambiguation using measures of word semantic similarity. In *International Conference on Semantic Computing (ICSC 2007)*, pages 363–369. IEEE, 2007.

[101] Ann E. Sizemore, Jennifer E. Phillips-Cremins, Robert Ghrist, and Danielle S. Bassett. The importance of the whole: Topological data analysis for the network neuroscientist. *Network Neuroscience*, 3(3):656–673, 07 2019. ISSN 2472-1751. doi: 10.1162/netn_a_00073.

[102] Daniel K. Sodickson and Warren J. Manning. Simultaneous acquisition of spatial harmonics (SMASH): Fast imaging with radiofrequency coil arrays. *Magnetic Resonance in Medicine*, 38(4):591–603, 1997. ISSN 07403194, 15222594. doi: 10.1002/mrm.1910380414.

[103] Bernadette J Stolz, Tegan Emerson, Satu Nahkuri, Mason A Porter, and Heather A Harrington. Topological data analysis of task-based fMRI data from experiments on schizophrenia. *Journal of Physics: Complexity*, 2(3):035006, may 2021. doi: 10.1088/2632-072x/abb4c6.

[104] Shuji Suzuki, Takashi Ishida, Ken Kurokawa, and Yutaka Akiyama. GHOSTM: A GPU-accelerated homology search tool for metagenomics. *PLOS ONE*, 7(5):1–8, 05 2012. doi: 10.1371/journal.pone.0036060.

[105] Shuji Suzuki, Masanori Kakuta, Takashi Ishida, and Yutaka Akiyama. GPU-acceleration of sequence homology searches with database subsequence clustering. *PLOS ONE*, 11(8):1–22, 08 2016. doi: 10.1371/journal.pone.0157338.

[106] Tadas Temčinas. Local homology of word embeddings. *arXiv:1810.10136*, 2018.

[107] Sergios Theodoridis. *Machine learning: a Bayesian and optimization perspective*. Academic Press, 2 edition, 2020. ISBN 978-0-12-818803-3.

[108] Cédric Villani. *Topics in optimal transportation*, volume 58 of *Graduate studies in mathematics*. American Mathematical Society, 2003. ISBN 978-0-8218-3312-4.

[109] Hubert Wagner, Paweł Dłotko, and Marian Mrozek. Computational topology in text mining. In *Computational Topology in Image Context*, pages 68–78. Springer, 2012.

[110] Ze Wang, Jiongjiong Wang, and John A Detre. Improved data reconstruction method for grappa. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 54(3):738–742, 2005.

[111] Wei Wei, EA Huerta, Bradley C Whitmore, Janice C Lee, Stephen Hannon, Rupali Chandar, Daniel A Dale, Kirsten L Larson, David A Thilker, Leonardo Ubeda, and others. Deep transfer learning for star cluster classification: I. application to the PHANGS–HST survey. *Monthly Notices of the Royal Astronomical Society*, 493(3): 3178–3193, 2020. doi: 10.1093/mnras/staa325.

[112] Rob Williams. Intel's core i7-980x extreme edition – ready for sick scores?: Mathematics: Sandra arithmetic, crypto, microsoft excel. *Techgage*, 2010. URL `http://techgage.com/article/intels_core_i7-980x_extreme_edition_-_ready_for_sick_scores/8`.

[113] Philip Wolfe. Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235, 1969. doi: 10.1137/1011036.

[114] Qiong Wu, Xiaoqi Huang, Adam J. Culbreth, James A. Waltz, L. Elliot Hong, and Shuo Chen. Extracting brain disease-related connectome subgraphs by adaptive dense subgraph discovery. *Biometrics*, pages 1–13, 2021. doi: 10.1111/biom.13537.

[115] Li Xiao, Junqi Wang, Peyman H. Kassani, Yipu Zhang, Yuntong Bai, Julia M. Stephen, Tony W. Wilson, Vince D. Calhoun, and Yu-Ping Wang. Multi-hypergraph learning-based brain functional connectivity analysis in fmri data. *IEEE Transactions on Medical Imaging*, 39(5):1746–1758, 2020.

[116] Tengyu Xu, Zhuoran Yang, Zhaoran Wang, and Yingbin Liang. Doubly robust off-policy actor-critic: Convergence and optimality. *arXiv:2102.11866*, 2021.

[117] Bernard Ycart. Extreme points in convex sets of symmetric matrices. *Proceedings of the American Mathematical Society*, 95(4):607–612, 1985.

[118] Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with ucb-based exploration. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 11492–11502. PMLR, 13–18 Jul 2020.

[119] Youjia Zhou, Archit Rathore, Emilie Purvine, and Bei Wang. Topological simplifications of hypergraphs, 2021.

[120] Xiaojin Zhu. Persistent homology: An introduction and a new text representation for natural language processing. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[121] Dongmian Zou, Radu Balan, and Maneesh Singh. On lipschitz bounds of general convolutional neural networks. *IEEE Transactions on Information Theory*, 66(3):1738–1759, 2019. doi: 10.1109/TIT.2019.2961812.

[122] Chen Zu, Yue Gao, Brent Munsell, Minjeong Kim, Ziwen Peng, Yingying Zhu, Wei Gao, Daoqiang Zhang, Dinggang Shen, and Guorong Wu. Identifying high order brain connectome biomarkers via learning on hypergraph. In *Machine Learning in Medical Imaging*, pages 1–9, Cham, 2016. Springer International Publishing. ISBN 978-3-319-47157-0.

[123] Chen Zu, Yue Gao, Brent Munsell, Minjeong Kim, Ziwen Peng, Jessica R. Cohen, Daoqiang Zhang, and Guorong Wu. Identifying disease-related subnetwork connectome biomarkers by sparse hypergraph learning. *Brain Imaging and Behavior*, 13(4):879–892, Aug 2019. ISSN 1931-7565. doi: 10.1007/s11682-018-9899-8.