RESEARCH STATEMENT
Nora Youngs, Harvey Mudd College

1. INTRODUCTION

Navigation is one of the most vital functions of the brain. Without the ability to explore new environments, remember previous routes and relative positions, and find our way home again, we would (quite literally) be lost. In 2014, the Nobel Prize in Medicine and Physiology was awarded for the discovery of place cells and grid cells, the neurons responsible for this ability. Both place and grid cells are found in the hippocampus, a brain region necessary for both memory storage and navigation. The hippocampus is also one of the first regions of the brain to show degeneration under Alzheimer’s disease, which is why those patients can become lost and disoriented easily. Now that these cells have been identified, it is critical to determine how they provide us with an internal map of our environment.

Understanding how the brain encodes information is one of the major questions of neuroscience research. In the field of neural coding, the relationship between stimulus and response, and the creation of a “dictionary” to relate the two, is one area which has been very frequently investigated [1, 2, 3]. The aforementioned Nobel Prize is given for work of this kind [4, 5]. From this perspective, the neuroscientist who records a place cell neuron which fires when a rat is in the northeast corner of his environment would then look at the data and assume that the rat was in that northeast corner each time that neuron was firing. What is not yet known, however, is how the brain puts this information together into a working map, without the dictionary of responses and the bird’s eye view the neuroscientist is privy to. We believe that much can be learned from investigating the structure of the neural code on its own. This question is also important from a coding theory perspective, as it addresses error-correcting properties and decoding efficiency [6]. Also, it has been shown that the structure of the neural code can reflect the topological structure of the stimulus space [7].

My research has thus far focused on the question motivated by this second viewpoint - what can be learned about the stimulus space from the intrinsic structure of a neural code? Given a set of neurons \( \{1, \ldots, n\} \), we define a neural code \( C \subset \{0, 1\}^n \) as a set of binary patterns of neural activity. Each codeword \( c = (c_1, \ldots, c_n) \) corresponds to a subset of neurons \( \text{supp}(c) = \{i \mid c_i = 1\} \) which were recorded to be ‘on’ simultaneously at some point. This type of code (without specific firing rates or times) is often referred to as a combinatorial neural code [8, 9]. One common way to obtain a neural code is with receptive fields. A receptive field is a map \( f_i: X \rightarrow \mathbb{R}_{\geq 0} \) from a space of stimuli, \( X \), to the average firing rate of a neuron \( i \) in response to each stimulus. Commonly, the set \( U_i \subset X \) where \( f_i \) takes on positive values is called a receptive field for the \( i \)th neuron. We obtain a neural code from a receptive field as follows:

**Definition.** Let \( X \) be a stimulus space and \( \mathcal{U} = \{U_1, \ldots, U_n\} \) a collection of receptive fields with \( U_i \subset X \). The receptive field (RF) code of \( \mathcal{U} \) is

\[
C(\mathcal{U}) \overset{\text{def}}{=} \{c \in \{0, 1\}^n \mid \left( \bigcap_{i \in \text{supp} \ c} U_i \right) \setminus \left( \bigcup_{j \notin \text{supp} \ c} U_j \right) \neq \emptyset \}.
\]

If \( X \subset \mathbb{R}^d \), and \( U_i \) is convex for all \( i \), then we say \( C(\mathcal{U}) \) is a convex receptive field code.

The particular example of receptive fields which motivates our work is these hippocampal place cells. As discussed, these are neurons whose receptive field is a 2-dimensional spatial region, the place field. These 2D regions overlap to create a set of smaller regions, each of which activates a particular set of neurons and thus has the same corresponding codeword (see figure at right).
Receptive field codes, however, are found in many other situations, such as orientation-selective neurons in visual cortex [10, 11], or those neurons which code for head direction. In any such situations, how can we extract the structure of the receptive fields from the neural code? What can we learn about the dimension of the underlying space? To make progress on these questions and others like them, we turn to algebraic geometry, which provides a natural framework for extracting topological and geometric structures from a space.

2. The Neural Ring

Although any code may be realized as a receptive field code in $\mathbb{R}$, there are codes which cannot be realized as convex receptive field codes in $\mathbb{R}^d$, for any $d$. Examples of this occur on as few as 3 neurons. The questions we hope to answer are the following: given a code $C$, is $C$ a convex receptive field code? If so, what is the minimal dimension $d$ for which we can realize this code as a receptive field code in $\mathbb{R}^d$? To work towards answers for these questions, we define an algebraic object called the neural ring. This, and much of the following section, is laid out in detail in our recent paper [12].

2.1. The neural ideal for general codes.

Definition. Given a code $C \subset \{0, 1\}^n$, we define the corresponding neural ideal $I_C \subset \mathbb{F}_2[x_1, ..., x_n]$ as

$$I_C \overset{\text{def}}{=} \{ f \in \mathbb{F}_2[x_1, ..., x_n] \mid f(c) = 0 \text{ for all } c \in C \}.$$  

That is, we define $I_C$ to be the set of polynomials which evaluate to 0 on all codewords: $C$ is exactly the variety $V(I_C)$. We define the neural ring to be $\mathbb{F}_2[x_1, ..., x_n]/I_C$, which is exactly the ring of functions $C \rightarrow \{0, 1\}$.

This abstract definition is a natural way to store the code, but in order to extract the combinatorial information that defines the code in a simpler way, we must look further. By finding a set of generators for $I_C$, we can understand the structure more concretely. To this end, define the ideal generated by the so-called Boolean relations $B = \langle \{x_i(1 - x_i) \mid i = 1, ..., n\} \rangle$. Then, for each $v \in \{0, 1\}^n$, define the polynomial $\rho_v = \prod_{i=1}^{n} x_i \prod_{v_i=0} (1 - x_i)$; note $\rho_v$ is essentially an indicator polynomial for $v$, i.e., $\rho_v(c) = 1 \iff c = v$. Define $J_C = \langle \rho_v \mid v \notin C \rangle$. Then, I have proven that we can decompose $I_C$ as follows:

Theorem. With $B$ and $J_C$ defined as above,

$$I_C = B + J_C.$$  

Here $B$, as the ideal generated by the Boolean relations, does not depend on $C$ at all. That is, because the code is binary, the polynomial $x_i(1 - x_i)$ will evaluate to 0 on any $v \in \{0, 1\}^n$. Thus, all the specific information in the code is contained within $J_C$. Note that we can define this ideal for any binary code $C$, not just a receptive field code. The generators of $J_C$ belong to a class of polynomials we have called pseudo-monomials. This list of pseudo-monomial generators for $J_C$ are often no simpler to deal with than the code itself; indeed, if the code has fewer than $2^{n-1}$ words, there are more generators for $J_C$ than there were codewords in $C$. Therefore, we would like to find a different set of generators of the same ideal. The most efficient choice we have found so far is a set of generators we refer to as the “canonical form” of $J_C$ (written $CF(J_C)$) made up of the minimal pseudo-monomials of $J_C$. The combinatorial information that defines the code is still contained within these generators, and for receptive field codes they have a nice interpretation.
2.2. The neural ideal for receptive field codes. Given a set \( U \) of receptive fields, we can encode their intersection information in an ideal:

**Definition.** Given a set of receptive fields, \( U = \{U_1 \ldots U_n\} \), define the corresponding ideal \( I_U \) as follows:

\[
I_U \overset{\text{def}}{=} \langle \prod_{i \in \sigma} x_i \prod_{j \in \tau} (1 - x_i) \mid (\bigcap_{i \in \sigma} U_i ) \subseteq (\bigcup_{j \in \tau} U_j) \rangle.
\]

Since the sets \( \sigma, \tau \) need not be disjoint, this seems different from the similarly-generated \( I_C \).

However, I proved the following useful theorem:

**Theorem.** If \( C = C(U) \) for \( U = \{U_1, \ldots, U_n\} \) a set of receptive fields, then \( I_C = I_U \).

By pulling out the trivial combinatorial information (e.g. that \( U_1 \subseteq U_1 \)) into \( B \), I then showed that we can interpret the canonical form to give minimal combinatorial information about the receptive fields:

**Theorem.** Let \( C \subset \{0,1\}^n \) be a neural code, and let \( U = \{U_1, \ldots, U_n\} \) be any collection of open sets (not necessarily convex) in a nonempty stimulus space \( X \) such that \( C = C(U) \). The canonical form of \( J_C \) is:

\[
J_C = \langle \{ x_{\sigma} \mid \sigma \text{ is minimal w.r.t. } U_\sigma = \emptyset \}, \ 
\{ x_{\sigma} \prod_{i \in \tau} (1 - x_i) \mid \sigma, \tau \neq \emptyset, \sigma \cap \tau = \emptyset, U_\sigma \neq \emptyset, \bigcup_{i \in \tau} U_i \neq X, \ \text{and } \sigma, \tau \text{ are each minimal w.r.t. } U_\sigma \subseteq \bigcup_{i \in \tau} U_i \}, \ 
\{ \prod_{i \in \tau} (1 - x_i) \mid \tau \text{ is minimal w.r.t. } X \subseteq \bigcup_{i \in \tau} U_i \} \rangle.
\]

We call the above three (disjoint) sets of relations comprising \( CF(J_C) \) the minimal Type 1 relations, the minimal Type 2 relations, and the minimal Type 3 relations, respectively.

The minimal pseudo-monomial generators that give the canonical form (and thus the minimal combinatorial information) can be found algorithmically from the code itself. A preliminary algorithm for this process was published in our original paper [12]. Since then, I have written Matlab code (since many neuroscience researchers use Matlab for their own data) which implements an improved version of the algorithm. This code takes as input a neural code and gives the canonical form for the ideal. This is done by producing the canonical form for a code consisting of a single codeword \( c \in C \), and then iteratively adding codewords, one at a time, until the final canonical form is reached. The improved version algorithm is described in detail in our forthcoming paper [13] and the Matlab code is available online [14].

As an example of the power of this algorithm, consider the following code on 5 neurons. After applying the algorithm as indicated in the workflow above, we get minimal information to help us draw the picture shown at right.

\[
C = \{00000, 10000, 01000, 00100, 00001, 11000, 10001, 01100, 00110, 00101, 00011, 11100, 00111\}.
\]
3. **Ongoing Work and Undergraduate Projects**

Our ongoing work in this area subdivides into three interrelated areas, described below. Each of these areas has questions which would make interesting and appropriate undergraduate research projects. The pace with which our ability to extract precise neural data is increasing, and the mathematics needed to analyze these vast data sets is not progressing at the same rate. Attracting students to this area is therefore vital, and research projects are an excellent way to pique students’ interest in the subject.

3.1. **Realizability.**

**Question.** How can we detect whether or not a given code $C$ is convex-realizable?

We know that some codes can be written as convex receptive field codes in $\mathbb{R}^d$ for some $d$, while others cannot. Some broad classes of convex receptive field codes are known (e.g. simplicial complex codes), but it is not yet certain how to detect for any code whether or not it is convex. Through work with collaborators, we have determined that there are local obstructions to convexity based on topological constraints. These local obstructions have a distinctive signature which can be recognized from the canonical form of the ideal. This is, however, only a partial result. This previous summer, an undergraduate research project run by one of my collaborators proved that these are not the only type of obstruction [15]; however, the complete set of obstructions is not yet known.

**Question.** If a given code $C$ is convex-realizable, how can we determine the minimal dimension $d$ so that $C$ is realizable in $\mathbb{R}^d$?

Some lower bounds are given by classical results such as Helly’s theorem [16] but as these bounds rely solely on the simplicial complex structure in the code, they are in many cases very loose lower bounds. For those codes which are known to be convex in $\mathbb{R}^2$, particularly place cell codes, we know that the canonical form of the neural ideal gives us all the information necessary to describe the accompanying layout of place fields. Initial work on local obstructions, as well as a sufficient condition for realizability in $\mathbb{R}^2$, are laid out in a forthcoming paper based on work at the AMS Mathematics Research Community in 2014 [17].

Questions in this area would be attractive to students interested in topology or convex geometry. This past summer, I worked with three Harvey Mudd undergraduates on the question of sparse codes, and they found necessary and sufficient conditions for realizability in 2-sparse codes [18] Much of our analysis still centers on the simplicial complex of the code, and simplicial complexes are important but relatively friendly topological objects. Future projects in this area might involve the student researching existing theorems in convex geometry or realizability of simplicial complexes, and applying their findings to obtain new bounds, restrictions, or classes of examples, as well as signatures of these obstructions or characteristics in the canonical form.

3.2. **Maps Between Codes.** The initial portion of my research focused on defining the neural ring for an individual code and documenting its attributes. From the perspective of data analysis it is also crucial to consider the natural relationships between codes and thus the corresponding relationships among the associated neural rings. Several natural maps from one code to another exist; for example, permutation of the indices, inclusion of one code into another with more codewords, or truncation to a shorter code. In order to address the previously stated open questions, we use these maps to see how important attributes of the codes (minimal embedding dimension, convex realizability) are changed by certain basic operations, and how these changes are reflected in the algebraic properties preserved by the corresponding maps. We consider this to be a vital step in characterizing realizability and dimension [13].

**Theorem.** The set of maps $\{q : C \to \mathcal{D}\}$ and the set of ring homomorphisms $\{\phi : \mathbb{R}_D \to \mathbb{R}_C\}$ are in bijection. This bijection is given by taking the pullback map: a code map $q$ corresponds to the ring homomorphism $\phi_q$, where $\phi_q(f) = q^* f$. 

4
However, considering the induced ring homomorphisms on the neural rings does not lead to satisfying conclusions. For example, two neural rings are isomorphic exactly when the two corresponding codes have the same number of codewords, regardless of the number of neurons of each code. Therefore, we consider the neural rings as modules under the ambient ring $R[n] = \mathbb{F}_2[x_1, \ldots, x_n]/B$ which allows us to retain the structure from the code by preserving the inherent correspondence between neuron $i$ and the variable $x_i$. We can now define a more relevant notion: a homomorphism $\tau : R[m] \rightarrow R[n]$ is linear-monomial if $\phi(y_i) \in \{x_j, 0, 1\}$. Then a ring homomorphism $\phi : R_D \rightarrow R_C$ is a neural ring homomorphism if there is some neuron-preserving ring homomorphism $\tau$ between the ambient rings such that $\phi(ry) = \tau(r)\phi(x)$.

**Theorem.** $\phi : R_C \rightarrow R_D$ is a neural ring homomorphism if and only if $q^*\phi$ is a composition of the following elementary code maps:

1. Deleting a neuron
2. Permutation of labels
3. Repeating a neuron
4. Adding a trivial (1 or 0) neuron
5. Adding codewords

**Question.** How do neuron-preserving code maps affect important properties, like realizability or dimension? How are these changes reflected by the neural ring homomorphism?

We have established five basic neuron-preserving code maps; now we need to determine what their compositions look like algebraically and geometrically. A student interested in algebra or convex geometry would try to answer questions like: Can we make an unrealizable code realizable by applying any of these maps? If a code is realizable, can we raise or lower the dimension by any number using one of these maps? What constructions would cause this to occur? What classes of examples have dimensions which are not affected by these maps? The tangible, example-driven nature of these questions make them approachable, even without extensive algebra background.

Questions also exist in this area for students with interest in neural data analysis. Considering maps between codes is our primary option thus far for dealing with error in our codes - if our code misses a codeword or a neuron, can we map it to the correct code? What would change? Students interested in neuroscience could determine from neuroscience literature what types of errors are most likely, and how these would change the properties of our codes.

3.3. **Data analysis.** As discussed, we have developed an algorithm which can take a code and generate the canonical form. However, we have yet to create an algorithm which creates a possible arrangement of sets which generate the code, even for small examples known to be realizable.

**Problem.** If a code is known to be realizable, creates a possible arrangement of convex open sets which realizes $C$.

Although this is a hard problem in general, we hope to make some progress restricting to small numbers of neurons or low dimensional examples. When analyzing an actual neural code from data, it is necessary to consider errors such as missing neurons or codewords, and how the structure changes if the errors are corrected. We hope that rigorous results from the analysis of maps between codes will result in large classes of known examples.

For undergraduate students with interest in programming, many aspects of this process would make interesting projects. During the last year of my PhD, I worked with an undergraduate student on optimizing the existing Matlab code, and creating partial realizations using Sage. Future projects include analyzing sparse codes, looking at the number of relations of each type occurring in the canonical form, and further work on algorithms to create arrangements of sets which realize as much of the code as possible. For those with interests in both algebra and programming, creating a version of this algorithm in Macaulay2 would cater to both their interests and the needs of the project.
REFERENCES


