Chapter 7

Analysis of Combinatorial Neural Codes: An Algebraic Approach

Carina Curto*, Alan Veliz-Cuba† and Nora Youngs‡
**Department of Mathematics, The Pennsylvania State University, University Park, PA, United States, †Department of Mathematics, University of Dayton, Dayton, OH, United States, ‡Department of Mathematics and Statistics, Colby College, Waterville, ME, United States*

7.1 INTRODUCTION

7.1.1 Biological Motivation: Neurons With Receptive Fields

A major challenge in neuroscience is to understand how the brain encodes such an enormous amount of information about the outside world. By observing the behavior of neurons in response to various stimuli, we can hypothesize about the function of groups of neurons, and try to create a dictionary which can predict how the brain will respond to different stimuli. However, to understand how the brain uses the neural response to stimuli to create its own picture of the outside world, we must consider how the firing of a population of neurons intrinsically encodes information about the space of possible stimuli.

One specific example of a population of neurons which provides an internal representation of an external structure is that of *place cells*, discovered by O'Keefe and Dostrovsky in the early 1970s [1]. Place cells are neurons which code for a particular region of an animal's (2D) spatial environment. In this example, the stimulus space is the set of possible locations in the environment. Each place cell is associated with a particular region of the environment (a *place field*); when the animal is within that region, the associated cell fires. Fig. 7.1 shows the activity of three different place cells as an animal passes through their respective place fields.

From an external perspective, determining the place field associated with a particular place cell is simply a matter of recording the activity of that cell while simultaneously observing the animal as it moves about its environment. However, for the animal itself to obtain useful information from the firing of

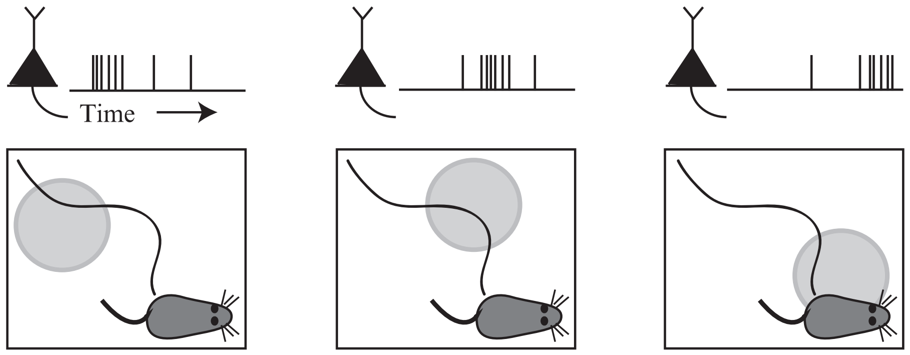

FIG. 7.1 Three place cells (represented by the *triangular* "neurons"), and the firing times of each one as an animal passes through their respective place fields (represented by *shaded gray regions*). *(Figure first published in C. Curto, What can topology tells us about the neural code?, Bull. AMS 54 (1) (2017) 63–78, © 2017 American Mathematical Society.)*

a place cell without the bird's eye view of the researcher, there would have to be some spatial information intrinsic to the firing of these cells. Considered individually, these cells give very little information about the shape of the animal's environment. However, by considering the collective behavior of the neurons, considerably more information can be determined [2].

Place cells are not the only neurons whose firing behavior can be associated with regions of stimulus. Other cells with such *receptive fields*, as they are called, are not uncommon; some examples include cells which code for angle of a visual stimulus, or 3D place fields in bats [3, 4]. With each of these examples, there is a space of stimulus in some dimension—one dimension for angles, and three dimensions for bat place fields. Given a set of any of these cells, we could record their firing behavior for various stimuli, obtaining what is called a (combinatorial) neural code.

A *neural code* $\mathcal{C}$ on n neurons is simply a subset of $\{0,1\}^n$ assumed to represent the behavior of the neurons in the set $\{1,\ldots,n\} = [n]$. We associate each codeword $\mathbf{c} \in \mathcal{C}$ to its support, representing the set of neurons which the codeword indicates were firing.

Definition 7.1. Let $\mathbf{c} \in \{0,1\}^n$. The *support* of $\mathbf{c}$ is the set $\mathrm{supp}(\mathbf{c}) = \{i \in [n] \mid \mathbf{c}_i = 1\}$. Likewise, we can associate the entire code $\mathcal{C}$ to the collection of subsets of neurons observed in the code: $\mathrm{supp}(\mathcal{C}) = \{\sigma \subset [n] \mid \sigma = \mathrm{supp}(\mathbf{c})$ for some $\mathbf{c} \in \mathcal{C}\}$.

The presence of the codeword $\mathbf{c}$ in $\mathcal{C}$ indicates that, at some point, the neurons in $\mathrm{supp}(\mathbf{c})$ were firing, while the other neurons were silent. Thus, each individual codeword indicates a set of neurons which were observed to respond a common stimulus; the entire code can be seen as providing a relatively complete picture of which combinations of neurons (and thus receptive fields) are possible within this particular set of neurons and receptive fields.

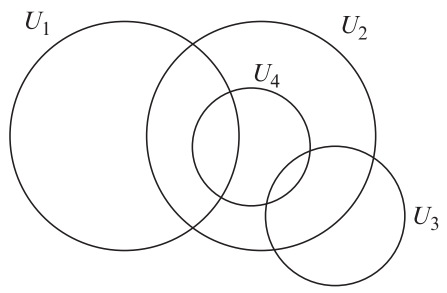

FIG. 7.2 A collection $\mathcal{U} = \{U_1, U_2, U_3, U_4\}$ of four receptive fields in $\mathbb{R}^2$.

Given knowledge of the receptive fields (and assuming that a broad range of stimuli in the space is experienced), we can predict the neural code associated with these receptive fields by considering the possible combinations of receptive fields a stimulus could come from.

Definition 7.2. Let X be a stimulus space. Given a collection of receptive fields $\mathcal{U} = \{U_1, \ldots, U_n\}$ with $U_i \subset X$ the receptive field for neuron i, we define the associated code $\mathcal{C}(\mathcal{U})$ as follows:

$$\mathcal{C}(\mathcal{U}) \overset{\text{def}}{=} \left\{ \mathbf{c} \in \{0,1\}^n \mid \left(\bigcap_{i \in \text{supp}(\mathbf{c})} U_i \right) \Bigg\backslash \left(\bigcup_{i \notin \text{supp}(\mathbf{c})} U_i \right) \neq \emptyset \right\}.$$

That is, $\mathcal{C}(\mathcal{U})$ encodes the set of nonempty regions formed by the sets in $\mathcal{U}$. If $\mathcal{C} = \mathcal{C}(\mathcal{U})$ for some collection of sets $\mathcal{U}$, we say $\mathcal{U}$ is a *realization* of $\mathcal{C}$ in the space X.

Example 7.1. Consider the collection of receptive fields $\mathcal{U} = \{U_1, U_2, U_3, U_4\}$ in $\mathbb{R}^2$ as represented in Fig. 7.2.

The code associated with this collection of sets is

$$\mathcal{C}(\mathcal{U}) = \{0000, 1000, 0100, 0010, 1100, 0110, 0101, 1101, 0111\};$$

and we can say that $\mathcal{U}$ is a realization of this code. Note that other realizations of the same code may exist.

To obtain the code associated with a known arrangement $\mathcal{U}$ of n receptive fields, we must simply determine which of the possible 2^n regions actually occur, and thus which combinations of neurons are possible. However, in a neural data context where the receptive fields are usually not known to us, we are more interested in the reverse question: given a set of neural data recorded in response to a certain type of stimulus, what can we say about the receptive fields, the relationships between them, and the structure of the space of stimuli they represent? We frame this question as follows:

Question 1. Given a neural code $\mathcal{C} \subset \{0, 1\}^n$, can we find a realization for it in a particular stimulus space X? If so, how can we find such a realization, and what can it tell us about the stimulus space X?

The next few sections will describe one approach to answering this question, using algebraic structures to encapsulate the information given by the code and extract relevant combinatorial features. These methods can be applied to any combinatorial code, not only to neural codes. However, in order to interpret this information to obtain meaningful results in the neural context, it will be necessary to make additional assumptions about the receptive fields U_i. We now briefly describe why this is so.

If no restrictions (e.g., convex, connected, etc.) of any kind are placed on the receptive fields U_i, then we can find a realization of any code, in nearly any space. In particular, there is no particular dimensional or structural information about the space associated with the code.

Exercise 7.1. Show that any arbitrary code $\mathcal{C}$ has a realization $\mathcal{U}$ in $\mathbb{R}$.

Such generic realizations are not satisfying for a neural code, as the receptive fields involved in the realization bear no resemblance to those seen in the neuroscience context and thus give no insight about the structure of the stimulus space. In Exercise 7.1, we see that without any assumptions on the receptive fields we could conclude every stimulus space is structurally equivalent to $\mathbb{R}$. However, by making a few reasonable assumptions about the nature of the receptive fields, we can drastically change the amount we can learn about the stimulus space from a neural code. Our main assumption will be to assume that the receptive fields U_i are *convex*; Section 7.4 gives more information about the results which can be obtained when we make this assumption. These results will rely heavily upon the information the code implies about the relationships between the receptive fields themselves.

7.1.2 Receptive Field Relationships

The neural code captures relationships between the firing behavior of different neurons, and thus captures ideas about the interactions between their receptive fields. For example, if there is a codeword $\mathbf{c} \in \mathcal{C}$ where $i, j \in \text{supp}(\mathbf{c})$ (so $\mathbf{c}_i = \mathbf{c}_j = 1$), we would note that at some point, neurons i and j are both firing. Thus their receptive fields overlap, or equivalently, $U_i \cap U_j$ is nonempty. However, if no such codeword appears, then we must assume that $U_i \cap U_j = \emptyset$. Similarly, we might assume that $U_i \subset U_j$ if we observe no codewords where neuron i is firing and neuron j is not. In general, a relationship of the form

$$\bigcap_{i\in\sigma} U_i \subset \bigcup_{j\in\tau} U_j$$

is referred to as a *receptive field (RF) relationship*, where we use the conventions that $\bigcap_{i\in\emptyset} U_i = X$, the entire stimulus space, and $\bigcup_{j\in\emptyset} U_j = \emptyset$.

Example 7.2. Consider the code $\mathcal{C} = \{000, 100, 010, 011\}$. Since there is no codeword where all three neurons fire together, we note the RF relationship $U_1 \cap U_2 \cap U_3 = \emptyset$ must hold in any realization of $\mathcal{C}$. We also note that there is no codeword where both neuron 1 and neuron 2 fire, so the RF relationship $U_1 \cap U_2 = \emptyset$ holds. Moreover, the former RF relationship we found is a consequence of the latter. Similarly, we note that neuron 3 fires only when neuron 2 is also firing, so $U_3 \subset U_2$ in any realization of $\mathcal{C}$.

Exercise 7.2. Extract as many receptive field relationships as you can from the following code:

$$\mathcal{C} = \{0000, 0001, 0011, 0111, 1001\}.$$

Are any of these RF relationships consequences of others on the list?

Importantly, the RF relationships extracted from a code show relationships which must hold in *any* realization of the code $\mathcal{C}$; they are features of the code, rather than of any one particular realization $\mathcal{U}$.

Exercise 7.3. Let $\mathcal{C}$ be the code from Exercise 7.2. Draw two different realizations of $\mathcal{C}$ in $\mathbb{R}^2$. Verify that the set of RF relationships you found in that exercise hold for both realizations.

While RF relationships are associated directly with a code $\mathcal{C}$, there are some RF relationships which will hold for *any* code $\mathcal{C}$, as the following exercise shows.

Exercise 7.4. Let $\mathcal{C}$ be an arbitrary code. Show that if σ and τ are subsets of $[n]$ such that $\sigma \cap \tau \neq \emptyset$, then any receptive field relationship of the form

$$\bigcap_{i\in\sigma} U_i \subset \bigcup_{j\in\tau} U_j$$

must be true in any realization $\mathcal{U}$ of $\mathcal{C}$.

We now give a preview of how RF relationships can be used to infer structure.

Example 7.3. Consider the code $\mathcal{C} = \{000, 010, 001, 110, 101\}$. We observe that in this code, neuron 1 never fires without one of neuron 2 or neuron 3 firing, so in any realization $\mathcal{U}$ of $\mathcal{C}$, we must have $U_1 \subset (U_2 \cup U_3)$. However, we also see that neurons 2 and 3 never fire at the same time, so in any realization, $U_2 \cap U_3 = \emptyset$. Thus, U_1 is contained within two completely disjoint sets.

If we make no assumptions about the properties of the receptive fields U_i, then we have merely made observations about the code itself which must hold in any realization. If, however, we make some assumptions about properties of the receptive fields, then we can determine much more interesting properties of possible realizations. For example, we may find that a realization exhibiting RF relationships is impossible.

Exercise 7.5. Show that if that our sets U_i must be convex and open, then there is no realization of the code $\mathcal{C} = \{000, 010, 001, 110, 101\}$ in $\mathbb{R}^2$ (or, indeed, in $\mathbb{R}^d$ for any d).

This exercise exhibits one example of a *topological obstruction*, a feature which indicates that this code cannot have a realization consisting of convex open sets. Section 7.4 goes into more detail about such obstructions.

For small examples, as in Exercise 7.2, we can list all the receptive field relationships by hand. However, for larger codes, such a list will not be feasible. Nor is it clear that a full list is necessary, since some of these relationships are redundant, or are true for every code, and thus convey no important information about the particular code in question. Thus, we need a method to efficiently extract the important RF relationships from a code.

For this, we will turn to algebraic geometry. We motivate its use by considering the example of the simplicial complex, a related combinatorial/topological object which has a well-known algebraic encoding.

7.1.3 The Simplicial Complex of a Code

As we have noted, a code $\mathcal{C}$ represents a collection $\text{supp}(\mathcal{C})$ of subsets of $[n]$. A simplicial complex is also a collection of subsets of $[n]$, but with an additional property.

Definition 7.3. An abstract *simplicial complex* Δ on $[n]$ is a collection Δ of subsets of $[n]$, such that whenever $\sigma \in \Delta$ and $\tau \subset \sigma$, we have $\tau \in \Delta$ also.

For a particular code $\mathcal{C}$, $\text{supp}(\mathcal{C})$ may not be a simplicial complex. However, we can associate a simplicial complex to any code as follows.

Definition 7.4. Define the *simplicial complex of the code* $\mathcal{C} \subset \{0, 1\}^n$ as

$$\Delta(\mathcal{C}) \overset{\text{def}}{=} \{\sigma \subset [n] \mid \sigma \subset \text{supp}(\mathbf{c}) \text{ for some } \mathbf{c} \in \mathcal{C}\}.$$

Exercise 7.6. Confirm that for any code $\mathcal{C}$, the set $\Delta(\mathcal{C})$ as defined earlier is actually a simplicial complex.

Exercise 7.7. Let $\mathcal{C}$ be the code from Example 7.1. Compute the simplicial complex $\Delta(\mathcal{C})$.

Exercise 7.8. Show that the following three codes have the same simplicial complex; that is, that $\Delta(\mathcal{C}_1) = \Delta(\mathcal{C}_2) = \Delta(\mathcal{C}_3)$.

- $\mathcal{C}_1 = \{0000, 1000, 1100, 1101, 1110\}$
- $\mathcal{C}_2 = \{0000, 1000, 0100, 1100, 1010, 0110, 1110, 1101\}$
- $\mathcal{C}_3 = \{0000, 1000, 0100, 1010, 0110, 1110, 1101\}$

As we see in Exercise 7.8, distinct codes may have the same simplicial complex, as long as any set of neurons which fires together in one code also fires together in the other. This co-firing information would be captured in a realization by noting that the respective receptive fields overlap. We now introduce the *nerve*, a set which encodes the set of nonempty intersections of any potential realization.

Definition 7.5. Let $\mathcal{U} = \{U_1, \ldots, U_n\}$ be a collection of sets in $\mathbb{R}^d$. The *nerve* of $\mathcal{U}$ is

$$\mathcal{N}(\mathcal{U}) = \left\{ \sigma \subset [n] \mid \bigcap_{i\in\sigma} U_i \neq \emptyset \right\}.$$

The following two exercises illustrate the relationship between the simplicial complex of a code $\mathcal{C}$ and the nerve of a realization $\mathcal{U}$ of $\mathcal{C}$.

Exercise 7.9. Let $\mathcal{U} = \{U_1, U_2, U_3, U_4\}$ be the collection of sets from Example 7.1. Compute the nerve $\mathcal{N}(\mathcal{U})$, and show that it is the same as the simplicial complex $\Delta(\mathcal{C})$ computed in Exercise 7.7.

Exercise 7.10. Show that for any code $\mathcal{C} \subset \{0,1\}^n$ and any realization $\mathcal{U}$ of $\mathcal{C}$, we have

$$\Delta(\mathcal{C}) = \mathcal{N}(\mathcal{U}).$$

We will now discuss an algebraic method for encoding a simplicial complex, and as such, a brief note about algebra is in order. In the rest of this chapter, $k[x_1, \ldots, x_n]$ will denote the ring of polynomials in the variables $x_1, \ldots, x_n$ with coefficients in the field k. An ideal is a nonempty subset I of $k[x_1, \ldots, x_n]$ that is closed under addition (if $a, b \in I$ then $a + b \in I$) and under multiplication by elements in $k[x_1, \ldots, x_n]$ (if $a \in I$ and $f \in k[x_1, \ldots, x_n]$, then $fa \in I$). We will also often refer to an ideal *generated by* a set A, which we will denote $I = \langle A \rangle$; this is the set of elements which can be written as finite combinations of elements in I. For more on these definitions, see for example [5, 6].

Simplicial complexes can be encoded algebraically through the Stanley-Reisner ideal. For a set $\sigma \subset [n]$, denote $x_\sigma = \prod_{i\in\sigma} x_i$. The *Stanley-Reisner ideal* over the field k for the simplicial complex Δ on $[n]$ is defined as

$$I_{\Delta^c} \overset{\text{def}}{=} \langle x_\sigma \mid \sigma \notin \Delta \rangle \subset k[x_1, \ldots, x_n].$$

For a more thorough treatment of the Stanley-Reisner ideal, see [7] or [8, Chapter 6]. As the Stanley-Reisner ideal encodes simplicial complexes, it can be used to capture the simplicial complex of a code and as Exercise 7.10 shows, it therefore implies certain intersection information for receptive fields.

Example 7.4. Let $\mathcal{C}$ be the code from Example 7.1. The Stanley-Reisner ideal of $\Delta(\mathcal{C})$ is $I_{\Delta(\mathcal{C})} = \langle x_1x_3 \rangle$. This captures the fact that $\{1,3\} \notin \Delta(\mathcal{C})$, as $\mathcal{C}$ does not exhibit any codewords where neurons 1 and 3 both fire. In any realization of $\mathcal{C}$, we would see the RF relationship $U_1 \cap U_3 = \emptyset$.

The simplicial complex and the nerve capture information about RF relationships of the form $\bigcap_{i\in\sigma} U_i = \emptyset$. However, neither structure captures the information about relationships which show containment; that is, RF relationships of the form $\bigcap_{i\in\sigma} U_i \subset \bigcup_{j\in\tau} U_j$.

Since the simplicial complex information is insufficient to understand a code, we will now introduce an algebraic structure similar to the Stanley-Reisner ideal which encodes *all* RF relationships of a code. This will be the *neural ideal*.

7.2 THE NEURAL IDEAL

Except where otherwise noted, the material in Sections 2 and 3 comes from [9], where the idea of the neural ideal was introduced. As we consider only binary codes, we will work over the field of two elements, $\mathbb{F}_2 = \{0, 1\}$ and in particular, over the polynomial ring $\mathbb{F}_2[x_1, \ldots, x_n]$.

Definition 7.6. Suppose I is an ideal of a polynomial ring $k[x_1, \ldots, x_n]$. The *variety* of I, denoted $V(I)$, is defined to be the set of common zeros of the elements of I; that is,

$$V(I) = \{\mathbf{v} \in k^n \mid f(\mathbf{v}) = 0 \text{ for all } f \in I\}.$$

For an ideal $I \subset \mathbb{F}_2[x_1, \ldots, x_n]$, the variety $V(I)$ is a subset of $\mathbb{F}_2^n$. The neural ideal for a code $\mathcal{C}$ is designed to be an ideal whose variety is precisely $\mathcal{C}$. Before introducing the neural ideal, we consider an important related ideal called the Boolean ideal.

Exercise 7.11. Show that the set of polynomials

$$\mathcal{B} = \{f \in \mathbb{F}_2[x_1, \ldots, x_n] \mid f(\mathbf{v}) = 0 \text{ for all } \mathbf{v} \in \mathbb{F}_2^n\}$$

forms an ideal, and furthermore that $V(\mathcal{B}) = \mathbb{F}_2^n$. $\mathcal{B}$ is called the *Boolean ideal*.

Remark 7.1. Since we will be working over $\mathbb{F}_2$ for the remainder of the chapter, we note that $1 = -1$ in $\mathbb{F}_2$, and thus that $1 + x_i = 1 - x_i$. We usually choose to use the latter notation.

Exercise 7.12. Show that $\mathcal{B} = \langle x_i(1-x_i) \mid i \in [n]\rangle$; that is, that $\mathcal{B}$ is generated by the set of Boolean polynomials $x_i(1 - x_i)$.

The next few examples and exercises provide additional practice with ideals and varieties in $\mathbb{F}_2[x_1, \dots, x_n]$.

Example 7.5. Consider the ideal $I = \langle f_1, f_2, f_3\rangle \subset \mathbb{F}_2[x_1, x_2, x_3, x_4]$ where $f_1 = (1-x_1)x_2, f_2 = x_1(1-x_3)$, and $f_3 = x_3x_4$. This ideal contains polynomials such as $h = x_1x_3x_4$ (since $h = x_1f_3$), $g = x_1x_2x_3 - x_2$ (since $g = 1f_1 + x_2f_2$), and $w = x_3^3x_4$ (since $w = x_3^2f_3$).

The variety of this ideal is the set of binary strings that satisfy the system of equations

$$(x_1 - 1)x_2 = 0, \quad x_1(x_3 - 1) = 0, \quad x_3x_4 = 0.$$

An example of such a binary string is 0001. To find all the elements of the variety we can proceed as follows: Since $x_i \in \mathbb{F}_2$, the first equation implies that $x_1 = 1$ or $x_2 = 0$. In the case $x_1 = 1$, the other equations become $x_3 = 1$ and $x_4 = 0$. This gives the elements 1010 and 1110. In the case $x_1 = 0$, then $x_2 = 0$ and $x_3x_4 = 0$, so this gives the elements 0000, 0001, and 0010. This shows that the variety of the ideal is $V = \{1010, 1110, 0000, 0001, 0010\}$. This variety can also be said to have elements of the form 1*10, 000*, 00*0, where the asterisk means that any value is allowed.

Exercise 7.13. Consider the ideal $I = \langle x_1(1-x_3), x_3x_4\rangle \subset \mathbb{F}_2[x_1, x_2, x_3, x_4]$. Compute all of the elements in the variety of I and compare with the variety found in Example 7.5.

Exercise 7.14. Consider the ideal $I = \langle x_1(1 - x_2), x_2(1 - x_3), x_1(1 - x_3)\rangle$. Show that $x_1(1 - x_3) \in \langle x_1(1 - x_2), x_2(1 - x_3)\rangle$ and use this to show that $I = \langle x_1(1 - x_2), x_2(1 - x_3)\rangle$. Compute all of the elements in the variety of I.

7.2.1 Definition of the Neural Ideal

We now define our main object of study, the neural ideal of a code, which we will associate to the code a collection of polynomials. As we will see, these polynomials can be used to extract information about the RF relationships associated with $\mathcal{C}$. Recall that since $\mathbb{F}_2 = \{0, 1\}$, a code $\mathcal{C}$ can be thought of as a subset of $\mathbb{F}_2^n$.

Let $\mathbf{v} \in \mathbb{F}_2^n$. The *characteristic polynomial* of $\mathbf{v}$ is the polynomial $\rho_{\mathbf{v}} \in \mathbb{F}_2[x_1, \dots, x_n]$ defined by

$$\rho_{\mathbf{v}} = \prod_{i \in \mathrm{supp}(\mathbf{v})} x_i \prod_{j \notin \mathrm{supp}(\mathbf{v})} (1 - x_j).$$

Exercise 7.15. Show that $\rho_{\mathbf{v}}(\mathbf{c}) = \begin{cases} 1 & \mathbf{v} = \mathbf{c} \\ 0 & \mathbf{v} \neq \mathbf{c}. \end{cases}$

We define two ideals associated with each neural code.

Definition 7.7. Let $\mathcal{C} \subset \mathbb{F}_2^n$. The *vanishing ideal* of $\mathcal{C}$ is defined as

$$I_{\mathcal{C}} = \{f \in \mathbb{F}_2[x_1, \ldots, x_n] \mid f(\mathbf{c}) = 0 \text{ for all } \mathbf{c} \in \mathcal{C}\}.$$

Informally, $I_{\mathcal{C}}$ is the set of polynomials in $\mathbb{F}_2[x_1, \ldots, x_n]$ which vanish on the entire code.

The *neural ideal* of the code is the ideal

$$J_{\mathcal{C}} = \langle \rho_{\mathbf{v}} \mid \mathbf{v} \in \mathbb{F}_2^n \backslash \mathcal{C} \rangle.$$

That is, $J_{\mathcal{C}}$ is the ideal generated by the characteristic polynomials of noncodewords.

While $J_{\mathcal{C}}$ and $I_{\mathcal{C}}$ are defined quite differently, they capture similar information. The following exercises illustrate the relationship between the neural ideal $J_{\mathcal{C}}$ and the vanishing ideal $I_{\mathcal{C}}$.

Exercise 7.16. Show that every element of $J_{\mathcal{C}}$ will vanish on the entire code, that is, show that $J_{\mathcal{C}} \subset I_{\mathcal{C}}$.

Exercise 7.17. Show that

$$I_{\mathcal{C}} = J_{\mathcal{C}} + \mathcal{B}.$$

Exercise 7.17 shows that the polynomials in $I_{\mathcal{C}}$ and $J_{\mathcal{C}}$ differ only by a polynomial in $\mathcal{B}$ that vanishes on every vector in $\mathbb{F}_2^n$.

Exercise 7.18. Prove that both $I_{\mathcal{C}}$ and $J_{\mathcal{C}}$ have the same variety, $\mathcal{C}$, that is,

$$V(I_{\mathcal{C}}) = V(J_{\mathcal{C}}) = \mathcal{C}.$$

We now give a specific example of the ideal $J_{\mathcal{C}}$ for a code $\mathcal{C}$, and note how the relationship between $J_{\mathcal{C}}$ and $I_{\mathcal{C}}$ can be used to learn about the RF structure of the code.

Example 7.6. Consider the code $\mathcal{C} = \{000, 010, 110, 011\}$. Then,

$$\begin{aligned} J_{\mathcal{C}} &= \langle \rho_v \mid v \in \{001, 100, 101, 111\} \rangle \\ &= \langle (1 - x_1)(1 - x_2)x_3, x_1(1 - x_2)(1 - x_3), x_1(1 - x_2)x_3, x_1x_2x_3 \rangle. \end{aligned}$$

This ideal has four generators, but one can reduce the number of generators needed as follows: Since the sum of the last two generators is x_1x_3, it is in $J_{\mathcal{C}}$. Similarly, by adding the first and third generators, we can see that $(1-x_2)x_3 \in J_{\mathcal{C}}$, and by adding the second and third we obtain that $x_1(1-x_2) \in J_{\mathcal{C}}$. It follows that $J_{\mathcal{C}} = \langle x_1x_3, (1-x_2)x_3, x_1(1-x_2)\rangle$ (see Exercise 7.19).

Beyond describing which codewords are not in $\mathcal{C}$, the polynomials in $J_{\mathcal{C}}$ tell us further information about the code. For example, we know the polynomial $x_1x_3 \in J_{\mathcal{C}}$. Since $J_{\mathcal{C}} \subset I_{\mathcal{C}}$, this means that for any $\mathbf{c} \in \mathcal{C}$, we have $c_1c_3 = 0$, and thus the first and third neurons never fire in the same codeword. This leads us to conclude that the RF relationship $U_1 \cap U_3 = \emptyset$ must hold for $\mathcal{C}$.

This example illustrates how we might find RF relationships from the polynomials in $J_{\mathcal{C}}$, and the following section will elaborate on this idea. This example also shows that while our definition for the neural ideal provides a specific set of generators, it is often possible to use a different, shorter list of polynomials as generators for $J_{\mathcal{C}}$. The following exercises illustrate this idea.

Exercise 7.19. Let $\mathcal{C}$ be as in Example 7.6, and let $K = \langle x_1x_3, (1-x_2)x_3, x_1(1-x_2)\rangle$. Show that $J_{\mathcal{C}} = K$ and that $V(K) = V(J_{\mathcal{C}}) = \mathcal{C}$.

Exercise 7.20. Show that the ideals $J = \langle x_1(1-x_2)x_3, x_1x_2x_3\rangle$ and $K = \langle x_1x_3\rangle$ are equal. Show that there is a unique code $\mathcal{C} \subset \mathbb{F}_2^3$ such that $J = J_{\mathcal{C}}$.

Exercise 7.21. Suppose $J_{\mathcal{C}} = \langle x_1(1-x_2), x_1x_2, (1-x_1)(1-x_2), (1-x_1)x_2\rangle$. Show that $J_{\mathcal{C}} = \langle 1\rangle$. What would the associated code $\mathcal{C} \subset \mathbb{F}_2^2$ be in this case?

Exercise 7.22. Consider the code $\mathcal{C} = \{000, 111, 011, 001\}$. Compute $J_{\mathcal{C}}$ and show that $x_1(1-x_3)$ is in $J_{\mathcal{C}}$.

Exercise 7.23. Consider the code $\mathcal{C} = \{000, 100, 110, 010, 011, 001, 101\}$. Compute $J_{\mathcal{C}}$.

Exercise 7.24. Consider the code $\mathcal{C} = \{000, 100, 110, 010, 011, 001\}$. Compute $J_{\mathcal{C}}$, and compare this result with the previous exercise.

7.2.2 The Neural Ideal and Receptive Field Relationships

Since the variety of both ideals $I_{\mathcal{C}}$ and $J_{\mathcal{C}}$ is precisely the code $\mathcal{C}$, information about the elements of the ideals $J_{\mathcal{C}}$ and $I_{\mathcal{C}}$ can be translated into information about the code itself. In particular, we can often conclude RF relationships hold by showing that certain polynomials exist in $I_{\mathcal{C}}$. Example 7.6 gave one example; the following exercise considers another.

Exercise 7.25. Suppose $\mathcal{C} \subset \mathbb{F}_2^n$ is a code and $x_1(1-x_2) \in J_{\mathcal{C}}$. Show that the RF relationship $U_1 \subset U_2$ must hold for $\mathcal{C}$.

We now formalize this connection between the polynomials in the ideal $I_{\mathcal{C}}$ and the RF relationships associated with $\mathcal{C}$. Throughout this and the following sections, we will use the convention that $U_\sigma = \bigcap_{i\in\sigma} U_i$ and $x_\sigma = \prod_{i\in\sigma} x_i$.

Theorem 7.1 (Lemma 4.2 from [9])**.** *Let $\mathcal{C} \subset \mathbb{F}_2^n$ be a neural code, and let $\mathcal{U} = \{U_1, \ldots, U_n\}$ be any collection of sets in a stimulus space X such that $\mathcal{C} = \mathcal{C}(\mathcal{U})$. Then, for any sets $\sigma, \tau \subset [n]$,*

$$x_\sigma \prod_{i\in\tau}(1-x_i) \in I_{\mathcal{C}} \Leftrightarrow U_\sigma \subset \bigcup_{i\in\tau} U_i.$$

Exercise 7.26. Use Exercises 7.12 and 7.17 to prove that if $\sigma \cap \tau \neq \emptyset$, then $x_\sigma \prod_{i\in\tau}(1-x_i) \in I_{\mathcal{C}}$. Apply Theorem 7.1 to give a new proof of Exercise 7.4.

The previous exercise shows that when $\sigma \cap \tau \neq \emptyset$, Theorem 7.1 is a consequence of the Boolean relationships implied by $\mathcal{B}$ and provides trivial information about receptive fields, and in particular, information which is not dependent on the actual code $\mathcal{C}$. Thus, we look to $J_{\mathcal{C}}$ to obtain the RF relationships specific to the code.

Proposition 7.1. *If $\sigma \cap \tau = \emptyset$, we have*

$$x_\sigma \prod_{i\in\tau}(1-x_i) \in J_{\mathcal{C}} \Leftrightarrow U_\sigma \subset \bigcup_{i\in\tau} U_i.$$

Assuming that $\sigma \cap \tau = \emptyset$, there are three major types[1] of relationships we observe, depending on whether σ or τ or neither is empty.

Monomial: $(\sigma \neq \emptyset, \tau = \emptyset)$: $\quad x_\sigma \in J_{\mathcal{C}} \Leftrightarrow \bigcap_{i\in\sigma} U_i = \emptyset$

Mixed monomial: $(\sigma \neq \emptyset, \tau \neq \emptyset)$: $\quad x_\sigma \prod_{j\in\tau}(1-x_j) \in J_{\mathcal{C}} \Leftrightarrow U_\sigma \subset \bigcup_{j\in\tau} U_j$

Negative monomial: $(\sigma = \emptyset, \tau \neq \emptyset)$: $\quad \prod_{j\in\tau}(1-x_j) \in J_{\mathcal{C}} \Leftrightarrow X \subset \bigcup_{j\in\tau} U_j$

We use the convention that $\bigcap_{i\in\emptyset} U_i = X$, the entire stimulus space, and that $x_\emptyset = 1$.

1. In previous work, such as [9], these have been referred to as Type 1, 2, and 3 relationships.

We usually assume that our codes contain the all-zero codeword $\mathbf{0}$, representing an instance when none of the neurons were firing. In this case, we find that negative monomial relationships are impossible.

Exercise 7.27. Show that if $\mathbf{0} \in \mathcal{C}$ then it is impossible to have any negative monomials in $J_{\mathcal{C}}$.

The monomial relationships give the same information as the Stanley-Reisner ideal; namely, they describe the simplicial complex of the code.

Exercise 7.28. Show that if $\mathcal{C}_1, \mathcal{C}_2 \subset \mathbb{F}_2^n$ are two codes such that $J_{\mathcal{C}_1}$ and $J_{\mathcal{C}_2}$ contain the same set of monomials, then $\Delta(\mathcal{C}_1) = \Delta(\mathcal{C}_2)$.

We now give some examples and exercises using Theorem 7.1 and its consequence Proposition 7.1 to find the nontrivial RF relationships associated with a code by using its neural ideal.

Example 7.7. Consider the neural ideal $J = \langle x_1x_2x_3, x_1(1-x_2)x_3\rangle$; Exercise 7.20 shows that $J = J_{\mathcal{C}}$ for $\mathcal{C} = \{000, 100, 010, 001, 110, 011\}$. The generator $x_1x_3(1-x_2)$ encodes the information that the intersection of the receptive fields of neuron 1 and neuron 3 is contained in the receptive field of neuron 2 (i.e., $U_1 \cap U_3 \subseteq U_2$). Similarly, the generator $x_1x_2x_3$ encodes that $U_1 \cap U_2 \cap U_3 = \emptyset$. These two pieces of information imply that $U_1 \cap U_3$ is contained in both U_2 and its complement; thus, $U_1 \cap U_3 = \emptyset$. Since this information is encoded by the polynomial x_1x_3, it follows that $x_1x_3 \in J$. Indeed, x_1x_3 is the sum of the two generators of J, and Exercise 7.20 shows that $J = \langle x_1x_3\rangle$.

Example 7.8. Consider the neural ideal $J = \langle x_1(1-x_2), x_2(1-x_3)\rangle \subset \mathbb{F}_2[x_1, x_2, x_3]\rangle$. The generator $x_1(1-x_2)$ encodes the information that $U_1 \subseteq U_2$ and $x_2(1-x_3)$ encodes $U_2 \subseteq U_3$. This implies that $U_1 \subseteq U_3$, so the polynomial $x_1(1-x_3)$ is in J (see Exercise 7.14).

Exercise 7.29. Identify the RF relationships of the code with ideal $J_{\mathcal{C}} = \langle x_1x_2x_3, (1-x_1)x_2x_3, (1-x_1)x_2(1-x_3), (1-x_1)(1-x_2)x_3\rangle$. Then, use basic set theory to show that the same RF relationships can be determined from the generators of the ideal $K = \langle (1-x_1)x_3, (1-x_1)x_2, x_2x_3\rangle$. Does $J_{\mathcal{C}} = K$?

Exercise 7.30. Identify the RF relationships of the code with ideal $J_{\mathcal{C}} = \langle x_1x_2x_3, (1-x_1)(1-x_2)x_3, x_1(1-x_2)x_3\rangle$. Then, use basic set theory to show that the same information can be determined from the generators of the ideal $K = \langle (1-x_2)x_3, x_1x_3\rangle$. Does $J_{\mathcal{C}} = K$?

7.3 THE CANONICAL FORM

As the preceding examples and exercises emphasize, it is often the case that the receptive field relationships we discover from polynomials in $I_{\mathcal{C}}$ or $J_{\mathcal{C}}$ give redundant information. In Example 7.7 and Exercise 7.20, once we know that x_1x_3 is in $J_{\mathcal{C}}$ and thus have the RF relationship $U_1 \cap U_3 = \emptyset$, we can infer through basic set theory that $U_1 \cap U_2 \cap U_3 = \emptyset$. Thus, the polynomial $x_1x_2x_3 \in J_{\mathcal{C}}$ provides redundant information about the receptive fields. This polynomial is also in a sense algebraically redundant, since once we know $x_1x_3 \in J_{\mathcal{C}}$, then we know that $x_1x_2x_3 \in J_{\mathcal{C}}$, as it is a multiple of x_1x_3.

We now introduce a set of polynomials which captures the important RF relationship information, while excluding as much as possible those polynomials which give redundant information.

Definition 7.8. A *pseudo-monomial* is a polynomial of the form $x_\sigma \prod_{j\in\tau}(1-x_j)$, where $\sigma \cap \tau = \emptyset$.

Observe that for any $\mathbf{v} \in \mathbb{F}_2^n$, the characteristic polynomial $\rho_{\mathbf{v}}$ is a pseudo-monomial, but that the Boolean polynomial $x_i(1-x_i)$ is not a pseudo-monomial. Monomials, mixed monomials, and negative monomials are all examples of pseudo-monomials.

Definition 7.9. Let $J \subset \mathbb{F}_2[x_1, \ldots, x_n]$ be an ideal. A pseudo-monomial $f \in J$ is *minimal* in J if there is no pseudo-monomial $g \in J$ with $\deg(g) < \deg(f)$ such that $f = gh$ for some $h \in \mathbb{F}_2[x_1, \ldots, x_n]$.

Definition 7.10. Let $\mathcal{C} \subset \mathbb{F}_2^n$ be a code, and let $J_{\mathcal{C}}$ be the associated neural ideal. The *canonical form* of $J_{\mathcal{C}}$, denoted $CF(J_{\mathcal{C}})$, is the set of all minimal pseudo-monomials in $J_{\mathcal{C}}$.

Example 7.9. Let $\mathcal{C}$ be as in Example 7.7, so $J_{\mathcal{C}} = \langle x_1(1-x_2)x_3, x_1x_2x_3\rangle$. The canonical form is $CF(J_{\mathcal{C}}) = \{x_1x_3\}$, and as we have seen, $J_{\mathcal{C}} = \langle x_1x_3\rangle$.

Exercise 7.31. Show that the canonical form for the neural ideal generates the neural ideal, that is, that

$$J_{\mathcal{C}} = \langle CF(J_{\mathcal{C}})\rangle.$$

The canonical form collects *all* of the minimal pseudo-monomials in $J_{\mathcal{C}}$; as such, the term "minimal" comes with a warning. There are many cases in which the set of all minimal pseudo-monomials is not the smallest possible set of pseudo-monomials which generate the ideal, as in the following example.

Example 7.10. Consider the code $\mathcal{C} = \{000, 111, 011, 001\}$. Then $J_{\mathcal{C}} = \langle (1-x_1)x_2(1-x_3), x_1(1-x_2)(1-x_3), x_1(1-x_2)x_3, x_1x_2(1-x_3)\rangle$. The canonical form is $CF(J_{\mathcal{C}}) = \{x_1(1-x_2), x_2(1-x_3), x_1(1-x_3)\}$, but note that $J_{\mathcal{C}} = \langle x_1(1-x_2), x_2(1-x_3)\rangle$ (see Exercise 7.14). Thus, the canonical form does not provide the smallest possible set of pseudo-monomial generators.

Exercise 7.32. Draw a realization of the code with ideal $J_{\mathcal{C}} = \langle x_1x_2x_3, (1-x_1)x_2x_3, (1-x_1)x_2(1-x_3), (1-x_1)(1-x_2)x_3\rangle$. Extract the RF relationships from the canonical form $CF(J_{\mathcal{C}}) = \{x_3(1-x_1), x_2(1-x_1), x_2x_3\}$ and note that these are sufficient to draw a realization (see Exercise 7.29).

Exercise 7.33. Draw a realization of the code with ideal $J_{\mathcal{C}} = \langle x_1x_2x_3, x_1(1-x_2)x_3, (1-x_1)(1-x_2)x_3\rangle$. Extract the RF relationships from the canonical form $CF(J_{\mathcal{C}}) = \langle (1-x_2)x_3, x_1x_3\rangle$, and note that these are sufficient to draw the realization (see Exercise 7.30).

As these exercises illustrate, the canonical form gives a simpler set of information than the original presentation of $J_{\mathcal{C}}$, but still implies all of the same RF relationships. Thus, rather than using $J_{\mathcal{C}}$ as originally presented, we usually compute only the canonical form for $J_{\mathcal{C}}$. The next sections will show two methods for finding the canonical form of a code.

7.3.1 Computing the Canonical Form

Computing the canonical form for the neural ideal code $\mathcal{C}$ can be done iteratively by codeword. Writing down the canonical form for the ideal consisting of no codewords is simple; thereafter, we add the codewords one at a time, updating the canonical form at each step.

Here, we describe an algorithm which takes the canonical form for a given code $\mathcal{C} \subset \{0,1\}^n$, and a codeword $\mathbf{c} \in \{0,1\}^n$, and outputs the canonical form for $\mathcal{C} \cup \{\mathbf{c}\}$. It is generally assumed that $\mathbf{c} \notin \mathcal{C}$ since otherwise the canonical form will not change; however, the success of the algorithm does not depend on this assumption. This algorithm can then be iterated to build the canonical form for a code.

Algorithm 7.1

```
Input: The canonical form $CF(J_{\mathcal{C}})$ for a code $\mathcal{C} \subset \{0,1\}^n$, and a word $\mathbf{c} \in \{0,1\}^n$.
Output: The canonical form $CF(J_{\mathcal{C}\cup\{\mathbf{c}\}})$.
Step 0: Initialize empty lists $L$, $M$, and $N$.
Step 1: For $f \in CF(J_{\mathcal{C}})$, if $f(\mathbf{c}) = 0$, add $f$ to $L$; otherwise, add $f$ to $M$.
    Repeat for all polynomials $f \in CF(J_{\mathcal{C}})$.
Step 2: For each polynomial $g \in M$ and each $1 \le i \le n$, if
(i)  neither $x_i$ nor $(1-x_i)$ divide $g$, and
```

(ii) $(x_i - \mathbf{c}_i)g$ is not a multiple of some polynomial in L,
then add $(x_i - \mathbf{c}_i)g$ to N. Repeat for all polynomials $g \in M$ and $1 \leq i \leq n$.
Step 3: Output $L \cup N$.

A brief explanation of the algorithm: the polynomials in the original canonical form $CF(J_{\mathcal{C}})$ will either vanish on the new codeword $\mathbf{c}$, or they will not. In Step 1, we sort the polynomials by this feature. If the polynomial f does vanish on $\mathbf{c}$, then f will remain in the new canonical form (add f to L); if not, we add f to the list M to potentially adjust f so that it will vanish on $\mathbf{c}$. In Step 2, we perform this adjustment to each f, multiplying by a linear term which will certainly vanish on $\mathbf{c}$; namely, $(x_i - \mathbf{c}_i)$. If the result of this multiplication is not a pseudo-monomial, or if it is a multiple of some pseudo-monomial in L and therefore redundant, then we will not retain it; otherwise, we add it to the list of new pseudo-monomials to add to $CF(J_{\mathcal{C}})$ (list N). To ensure that we obtain the entire new canonical form and not only a subset, it is necessary to consider all such linear terms $(x_i - \mathbf{c}_i)$ and add all the valid possible adjustments to the new canonical form. Finally, we output the collection of the old polynomials which are still valid (list L) and the newly-created polynomials (list N) as the new canonical form.

A proof that this process will result in the correct canonical form for the new code can be found in [10].

Example 7.11. Let $\mathcal{C} = \{000, 100, 010\}$. Observe that in this code we have the RF relationships $U_3 = \emptyset$ since neuron 3 never fires, and $U_1 \cap U_2 = \emptyset$ since neurons 1 and 2 never fire together. The canonical form of $J_{\mathcal{C}}$ is $CF(J_{\mathcal{C}}) = \{x_1x_2, x_3\}$.

We will compute $CF(\mathcal{C}')$, where $\mathcal{C}' = \mathcal{C} \cup \{110\}$, inputting $CF(\mathcal{C}) = \{x_1x_2, x_3\}$ and $\mathbf{c} = 110$ into Algorithm 7.1:

Step 0: Set $L = M = N = \emptyset$.
Step 1 for x_3, x_1x_2: Since $f = x_3$ satisfies $f(110) = 0$, add x_3 to L. Since $f = x_1x_2$ does not satisfy $f(110) = 0$, add x_1x_2 to M. Then, $M = \{x_1x_2\}$ and $L = \{x_3\}$.
Step 2 for x_1x_2 and $i = 1, 2, 3$: It is not true that neither x_1 nor $1 - x_1$ divide x_1x_2. So N does not change.

It is not true that neither x_2 nor $1 - x_2$ divide x_1x_2. So N does not change.
It is true that neither x_3 nor $1 - x_3$ divide x_1x_2. However, $(x_3 - 0)x_1x_2$ is a multiple of a polynomial in L. So N does not change.

Step 3: The output is $L \cup N = \{x_3\}$. This ends Algorithm 7.1.

Thus, we find that $CF(\mathcal{C}') = \{x_3\}$. This result makes sense, since our new code $\mathcal{C}' = \{000, 100, 010, 110\}$ is characterized by the RF relationship $U_3 = \emptyset$.

Exercise 7.34. Let $\mathcal{C}' = \{000, 100, 010, 110\}$ as in Example 7.11. Define $\mathcal{C}'' = \mathcal{C}' \cup \{011\}$. Use Algorithm 7.1 to show that $CF(\mathcal{C}'') = \{x_1x_3, x_3(1 - x_2)\}$.

Algorithm 7.1 shows how to adjust a given canonical form to include one new codeword; computing the canonical form for a given code $\mathcal{C}$ is simply an iteration of this process. We note that the canonical form for the empty code $\mathcal{C}_0 = \emptyset$ is $CF(J_{\mathcal{C}_0}) = \{1\}$. Then, we order our code $\mathcal{C}$, and add in the codewords one at a time, using Algorithm 7.1 each time to compute the new canonical form. Once all codewords have been added, the result is the canonical form for the complete code $\mathcal{C}$. Algorithm 7.2 describes this process.

Algorithm 7.2

Input: A code $\mathcal{C} \subset \{0, 1\}^n$.
Output: The canonical form $CF(J_{\mathcal{C}})$.
Step 0: Arbitrarily order the codewords of $\mathcal{C}$ (so $\mathcal{C} = \{\mathbf{c}^1, \mathbf{c}^2, \dots, \mathbf{c}^d\}$).
Step 1: Define the code $\mathcal{C}_0 = \emptyset$. The canonical form of $\mathcal{C}_0$ is $CF(J_{\mathcal{C}_0}) = \{1\}$. Set $j = 1$.
Step 2: Define $\mathcal{C}_j = \mathcal{C}_{j-1} \cup \{\mathbf{c}^j\}$. Input $CF(J_{\mathcal{C}_{j-1}})$ and $\mathbf{c}^j$ into Algorithm 7.1; the output will be $CF(J_{\mathcal{C}_j})$. If $j = d$, output $CF(J_{\mathcal{C}_d})) = CF(J_{\mathcal{C}})$. Otherwise, set $j = j + 1$ and repeat Step 2.

Exercise 7.35. Using the iterative process described in Algorithm 7.2 and $\mathcal{C} = \{000, 100, 010\}$ as in Example 7.11, show that $CF(J_{\mathcal{C}}) = \{x_3, x_1x_2\}$. Then, reorder the codewords in a new way, recompute $CF(J_{\mathcal{C}})$, and show that the result is the same.

7.3.2 Alternative Computation Method: The Primary Decomposition

One well-known way to break down an ideal into manageable computational pieces is by finding its *primary decomposition*. This strategy has been leveraged with considerable success for the Stanley-Reisner ideal; see Ref. [8, Chapter 6] for a description. A *primary decomposition* of the neural ideal is

$$J_{\mathcal{C}} = \mathfrak{p}_1 \cap \cdots \cap \mathfrak{p}_r,$$

where each ideal $\mathfrak{p}_i$ is a primary ideal. We will not go into much detail about primary decompositions here, but we note that the primary decomposition of the neural ideal provides an alternative method to compute the canonical form, as well as providing combinatorial information about the code in its own right.

The primary decomposition gives information about the code which is complementary to that provided by the canonical form. Each ideal in the primary decomposition gives an interval in the Boolean lattice $\mathbb{F}_2^n$ covered by the code; on the other hand, the polynomials in the canonical form can be interpreted to describe intervals of *non*-codewords in the Boolean lattice. The primary ideals used in the decomposition described earlier can be written as ideals with linear generators; recombining these linear generators in their various possible combinations gives us the canonical form. This algorithm and the Boolean lattice interpretation are described in detail in [9].

Example 7.12. Consider the code $\mathcal{C} = \{000, 100, 010, 001, 101\}$. The neural ideal for this code is

$$J_{\mathcal{C}} = \langle (1 - x_1)x_2x_3, x_1x_2(1 - x_3), x_1x_2x_3 \rangle.$$

The canonical form of $J_{\mathcal{C}}$ is $CF(J_{\mathcal{C}}) = \{x_1x_2, x_2x_3\}$, where the pseudo-monomials correspond to the intervals $\{110, 111\}$ and $\{011, 111\}$ among the noncodewords. A primary decomposition of $J_{\mathcal{C}}$ is $J_{\mathcal{C}} = \langle x_1, x_3 \rangle \cap \langle x_2 \rangle$, where $\langle x_1, x_3 \rangle$ has as its variety the Boolean lattice interval $\{000, 010\}$ of codewords, and $\langle x_2 \rangle$ has the interval $\{000, 100, 001, 101\}$ as its variety. Note that polynomials obtained by taking one generator from each ideal in the primary decomposition result in precisely the generators of the canonical form.

7.3.3 Sage Code for Computations

As we have shown, it is possible to compute the canonical form by hand, but for large examples it quickly becomes tedious. Fortunately, the iterative algorithm we have described has been coded for computation in SageMath [11]. To use this code, first download the packages for neural computations from https://github.com/e6-1/NeuralIdeals. If necessary, rename the folder `NeuralIdeals-master` as `NeuralIdeals`. The following example will describe how to use the code to compute the canonical form and extract the RF relationships. For a more detailed guide, see [10].

Once the packages have downloaded, run the following to compile and load them.

```
load("NeuralIdeals/iterative_canonical.spyx")
load("NeuralIdeals/neuralcode.py")
load("NeuralIdeals/examples.py")
```

The code above should give the message

```
Compiling ./NeuralIdeals/iterative_canonical.spyx...
```

Example 7.13. We will use the Sage code to compute the canonical form for the neural code $\mathcal{C} = \{100, 010, 001, 101, 011, 111, 000\}$.

To define the neural code, run the following:

```
neuralCode = NeuralCode(['100','010','001','101','
             011','111','000'])
```

To compute the canonical form, run `CF=neuralCode.canonical()` and `CF` to see the result. This will give the following output.

```
Ideal (x0*x1*x2+x0*x1) of Multivariate Polynomial  Ring in
x0, x1, x2 over Finite Field of size 2
```

To have an output with factored generators, we run `CF=neuralCode.factored_canonical()` and `CF` to see the result. The output is `[(x2 + 1) * x1 * x0]`.

Furthermore, we can find the receptive field structure by running the following.

```
RF=neuralCode.canonical_RF_structure()
```

`RF` This prints the output, which in this case will be:

```
Intersection of U_['1', '0'] is a subset of Union of
   U_['2']
```

Thus, we obtain the following receptive field structure: $(U_0 \cap U_1) \subseteq U_2$. A possible receptive field in $\mathbb{R}^2$ is shown (on the left) in Fig. 7.3.

Note that we can also obtain a one-dimensional realization using the intervals $U_0 = (0,3), U_1 = (2,5)$, and $U_2 = (1,4)$.

Example 7.14. We will compute the canonical form of $\mathcal{C} = \{1010, 1110, 0000, 0001, 0010\}$ and attempt to draw a convex realization. We proceed as in the previous example.

```
neuralCode = NeuralCode(['1010','1110','0000','0001','
0010'])
CF=neuralCode.factored_canonical()
CF
```

This will give the output

```
[x1*(x0+1),(x2+1)*x0,(x2+1)*x1,x3*x0,x3*x1,x3*x2]
```

To obtain the receptive field structure we use `neuralCode.canonical_RF_structure()` and obtain

```
Intersection of U_['1'] is a subset of Union of U_['0']
Intersection of U_['0'] is a subset of Union of U_['2']
Intersection of U_['1'] is a subset of Union of U_['2']
Intersection of U_['3', '0'] is empty
Intersection of U_['3', '1'] is empty
Intersection of U_['3', '2'] is empty
```

Thus, we obtain the following neural field structure: $U_1 \subseteq U_0 \subseteq U_2$ and $U_3 \cap U_0 = U_3 \cap U_1 = U_3 \cap U_2 = \emptyset$. A possible receptive field in $\mathbb{R}^2$ is shown (on the right) in Fig. 7.3.

Note that we can also obtain a one-dimensional realization using the intervals $U_0 = (1,4), U_1 = (2,3), U_2 = (0,5)$, and $U_3 = (6,7)$.

Exercise 7.36. For the following codes, use Sage to compute the canonical form of $\mathcal{C}$. Use the RF relationships you obtain to draw a realization of $\mathcal{C}$, using convex sets if possible.

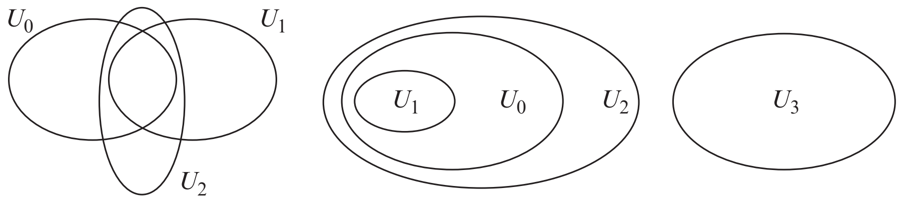

FIG. 7.3 A realization of the code in Example 7.13 (*left*), and one of the code in Example 7.14 (*right*).

1. $\mathcal{C} = \{000, 010, 110, 011\}$
2. $\mathcal{C} = \{000, 100, 110, 010, 011, 001, 101\}$
3. $\mathcal{C} = \{000, 100, 110, 010, 011, 001\}$
4. $\mathcal{C} = \{000, 111, 011, 001\}$
5. $\mathcal{C} = \{000, 100, 010, 101, 011\}$
6. $\mathcal{C} = \{0000, 1000, 0100, 1100, 0010, 1010, 0110, 1110, 0011, 0001\}$
7. $\mathcal{C} = \{0000, 1000, 1100, 0100, 0110, 0010, 0011, 0001\}$
8. $\mathcal{C} = \{0000, 1000, 0100, 1100, 0010, 1010, 0110, 1110, 0001, 1001, 0101, 1101, 0011, 1011, 0111\}$
9. $\mathcal{C} = \{0000, 1000, 0100, 1010, 0110, 0010, 0011\}$

Note: Some of the computations for this book chapter were done using the Ohio Supercomputer Center [12].

7.4 APPLICATIONS: USING THE NEURAL IDEAL

7.4.1 Convex Realizability

One of the main questions motivating the construction of the neural ideal is to determine which codes have realizations using *convex* sets, and which do not. Many types of neurons with receptive fields, including both 2D and 3D place cells, have natural convex receptive fields. Structural features of the code are known which either guarantee or prohibit the existence of a realization with convex sets, and the neural ideal and canonical form can often be used to detect these features. We will refer to a code which has a realization using open convex receptive fields in $\mathbb{R}^d$ for some d as a *convex* code.

As Exercise 7.8 shows, the simplicial complex alone is insufficient to characterize a code. In addition to the fact that different codes may have the same simplicial complex, codes with the same simplicial complex may have very different properties with respect to convex realizability.

Exercise 7.37. Let $\mathcal{C}_1, \mathcal{C}_2$, and $\mathcal{C}_3$ be the three codes from Exercise 7.8. Show that $\mathcal{C}_1$ has a realization with convex open sets in $\mathbb{R}$, that $\mathcal{C}_2$ does not have such a realization, but can be realized with convex open sets in $\mathbb{R}^2$, and that $\mathcal{C}_3$ cannot be realized with convex open sets in $\mathbb{R}^d$ for any dimension d.

As seen in Exercise 7.37 and previously in Exercise 7.5, there exist codes which are not convex. The obstructions to convexity in these exercises are topological in nature, and are specific examples of a broad class of obstructions known as *local obstructions*, defined in [13, 14]. These obstructions follow from an application of the Nerve lemma.

Lemma 7.1 (Nerve Lemma)**.** *If the sets $\mathcal{U} = \{U_1 \ldots, U_n\}$ are convex, then the homotopy type of $\bigcup_{i=1}^n U_i$ is equal to the homotopy type of the nerve $\mathcal{N}(\mathcal{U})$. In particular, $\bigcup_{i=1}^n U_i$ and $N(\mathcal{U})$ have exactly the same homology groups.*

The Nerve lemma is a consequence of [15, Corollary 4G.3]. Local obstructions arise in instances where features of the code dictate that any realization of the code using convex sets would violate the Nerve lemma; thus, a code which can be realized with convex sets must have no local obstructions.

Determining if a code $\mathcal{C}$ has local obstructions to convexity can be reduced to the question of determining whether $\mathcal{C}$ contains a particular minimal code, as Theorem 7.2 will show. However, we first require a definition.

Definition 7.11. Let Δ be a simplicial complex. For any $\sigma \in \Delta$, the *link* of σ in Δ is

$$\mathrm{Lk}_\sigma(\Delta) \overset{\text{def}}{=} \{\omega \in \Delta \mid \sigma \cap \omega = \emptyset \text{ and } \sigma \cup \omega \in \Delta\}.$$

Example 7.15. Consider the simplicial complex

$$\Delta = \{\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}, \{3,4\}, \{1,2,3\}\}.$$

Then, $\mathrm{Lk}_{\{1\}}(\Delta) = \{\emptyset, \{2\}, \{3\}, \{4\}, \{2,3\}\}$, whereas $\mathrm{Lk}_{\{2\}}(\Delta) = \{\emptyset, \{1\}, \{3\}, \{1,3\}\}$ and $\mathrm{Lk}_{\{1,3\}}(\Delta) = \{\emptyset, \{2\}\}$ (See Fig. 7.4).

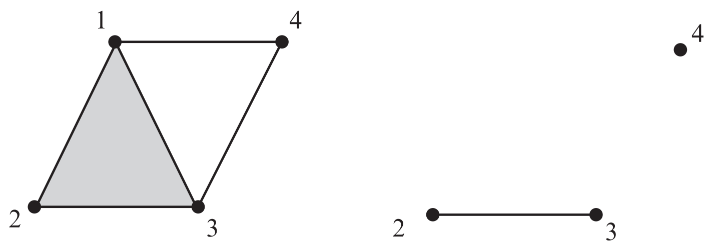

FIG. 7.4 The simplicial complex Δ from Example 7.15 (at *left*) and the link $\mathrm{Lk}_{\{1\}}(\Delta)$ (at *right*). Here, $\mathrm{Lk}_{\{1\}}(\Delta)$ is disconnected, and thus not contractible.

The necessary condition for a code to have no local obstructions will involve determining whether links are contractible. Informally, a topological space (such as a simplicial complex) is *contractible* if it is connected and has no "holes"; more formally, a space is contractible if it is homotopy-equivalent to a single point. In the previous example, $\mathrm{Lk}_{\{1\}}(\Delta)$ is disconnected, and thus is not contractible, but the other two links are both contractible.

Theorem 7.2 (Theorem 1.3 from [13]). *For each simplicial complex Δ, there is a unique minimal code $\mathcal{C}_{\min}(\Delta)$ with the following properties:*

1. *the simplicial complex of $\mathcal{C}_{\min}(\Delta)$ is Δ, and*
2. *any code $\mathcal{C}$ satisfying $\Delta(\mathcal{C}) = \Delta$ has no local obstructions if and only if $\mathcal{C} \supseteq \mathcal{C}_{\min}(\Delta)$.*

Determining this minimal code depends only on the simplicial complex $\Delta(\mathcal{C})$:

$$\mathcal{C}_{\min}(\Delta) = \{\sigma \in \Delta \mid \mathrm{Lk}_\sigma(\Delta) \text{ is noncontractible}\} \cup \{\emptyset\}.$$

Exercise 7.38. Recall code $\mathcal{C} = \mathcal{C}_3$ from Exercises 7.8 and 7.37. With $\Delta = \Delta(\mathcal{C})$ as computed in that exercise, show that $\mathrm{Lk}_{\{1,2\}}(\Delta)$ is not contractible. Conclude that $1100 \in \mathcal{C}_{\min}$, and that since $\mathcal{C}$ does not contain 1100, it cannot be convex.

Exercise 7.39. Recall $\mathcal{C} = \{000, 010, 001, 110, 101\}$, the code from Exercise 7.5. Find $\Delta(\mathcal{C})$. Then, identify a set σ such that $\mathrm{Lk}_\sigma(\Delta)$ is not contractible, and the associated codeword $\mathbf{c}$ with $\mathrm{supp}(\mathbf{c}) = \sigma$ is not in $\mathcal{C}$. Conclude that $\mathcal{C}$ is not convex.

While it can be difficult in general to detect these local obstructions from the canonical form, it is sometimes possible. Exercise 7.38 exhibits a simple example of such a situation.

Example 7.16. The code $\mathcal{C}_3$ from Exercises 7.8 and 7.38 has canonical form

$$CF(J_{\mathcal{C}_3}) = \{x_3x_4, x_1x_2(1-x_3)(1-x_4), x_3(1-x_1)(1-x_2), x_4(1-x_2), x_4(1-x_1)\}.$$

The first two polynomials together tell us that $(U_1 \cap U_2) \subset (U_3 \cup U_4)$ and $U_3 \cap U_4 = \emptyset$. This gives us a clue that we may have an obstruction using $\sigma = \{1, 2\}$, since in any convex realization, $U_1 \cap U_2$ should also be convex and thus connected. However, we just found that it is contained in the union of two disjoint sets, so it cannot be connected.

We can show that examples such as this one provide a pattern for similar local obstructions.

Exercise 7.40. Suppose $\mathcal{C} \subset \mathbb{F}_2^n$. Show that if there is some set $\sigma \subset [n]$ such that both of the following conditions hold:

1. $x_\sigma(1-x_i)(1-x_j) \in CF(J_\mathcal{C})$
2. $x_\sigma x_i x_j \in J_\mathcal{C}$

then $\mathcal{C}$ is not convex, as we can use σ to find a local obstruction.

Demonstrating that a code has a local obstruction proves immediately that the code cannot be convex. However, the converse does not hold; there are codes which have no local obstructions but still have no open convex realization, as in the following example.

Example 7.17 (Lienkaemper et al. [16]). The following code on five neurons has no local obstructions, but has no open convex realization:

$$C = \{00000, 00100, 00010, 10100, 10010, 01100, 00110, 00011, 11100, 10110, 10011, 01111\}.$$

The obstruction to convexity in this case is more geometric in its flavor—one can use convexity show that the straight line which connects points in two different regions has only a limited set of possible regions it can pass through, and only in certain orders, and this can be shown to be impossible in all cases.

Thus, local obstructions can only prove nonconvexity; we cannot assume that a code without local obstructions is necessarily convex. That said, several large classes of codes are known to have convex realizations. We will now show some examples of these classes, and give algebraic signatures from the canonical form which can tell us immediately if a code fits into one of these classes. The first such class is simplicial complexes.

Theorem 7.3 (Curto et al. [13]). *If $\mathcal{C}$ is a simplicial complex, then $\mathcal{C}$ has a convex realization.*

Proposition 7.2. *Let $\mathcal{C}$ be a neural code. Then $\mathcal{C}$ is a simplicial complex if and only if $CF(J_\mathcal{C})$ consists only of monomials.*

Exercise 7.41. Prove Proposition 7.2.

Theorem 7.3 and Proposition 7.2 combined show that by computing the canonical form $CF(J_\mathcal{C})$, we can immediately detect if $\mathcal{C}$ is a simplicial complex, and if so, conclude that $\mathcal{C}$ is convex. However, it is quite specific to require that a code be a simplicial complex, and many codes which are quite clearly realizable (including any code where one receptive field is covered by others, as in Example 7.1) are not simplicial complexes. Fortunately, there is a relaxation of the simplicial complex condition which is also known to be convex.

Definition 7.12. A code $\mathcal{C}$ is *intersection-complete* if for any $\mathbf{c}, \mathbf{d} \in \mathcal{C}$, there is a codeword $\mathbf{v} \in \mathcal{C}$ such that $\operatorname{supp}(\mathbf{v}) = \operatorname{supp}(\mathbf{c}) \cap \operatorname{supp}(\mathbf{d})$.

Exercise 7.42. Show that codes which are simplicial complexes are intersection-complete. Give an example to show that the converse does not hold.

The larger class of intersection-complete codes is also known to be convex realizable.

Theorem 7.4 (Cruz et al. [17])**.** *If $\mathcal{C}$ is intersection-complete, then $\mathcal{C}$ has a convex realization.*

Intersection-complete codes are more general than simplicial complex codes, but we can use the canonical form to detect them as well, as the following result indicates.

Theorem 7.5 (Curto et al. [18])**.** *A code $\mathcal{C}$ containing the all-zero word $\mathbf{0}$ is intersection-complete if and only if $CF(J_{\mathcal{C}})$ consists only of monomials, and mixed monomials of the form*

$$\prod_{i \in \sigma} x_i(1 - x_j).$$

From the previous two results, we see that codes whose canonical forms are entirely monomial (as with simplicial complexes) or allow only the most basic mixed monomials (intersection-complete codes) are convex. This might lead us to suspect that having mostly monomial-like relationships is somehow necessary for convexity. However, this is not the case, as codes with no monomials at all in their canonical form are also convex.

Theorem 7.6 (Curto et al. [13])**.** *If $\mathcal{C}$ is a neural code and $CF(J_{\mathcal{C}})$ contains no monomials, then $\mathcal{C}$ has a convex realization.*

Furthermore, codes with no monomial relationships are also known to have convex realizations in very low dimensions (1 or 2); see Fig. 7.5 and related discussion. This property also has a simple characterization in terms of the canonical form, as seen in the following exercise.

Exercise 7.43. Show that $CF(J_{\mathcal{C}})$ contains no monomials if and only if $\mathcal{C}$ contains the codeword $\mathbf{1} = 111\ldots1$.

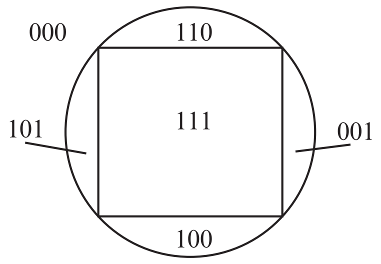

FIG. 7.5 Constructing a 2D realization of $\mathcal{C} = \{000, 100, 001, 101, 110, 111\}$. The codeword 111 is placed at the center of a polygon inscribed within a *circle*; the remaining codewords are assigned to the spaces created around the edges. We then take U_i to be the union of the regions labeled by codewords with neuron i firing.

7.4.2 Dimension

The question of realizability in general can tell us which codes can be represented with convex receptive fields in some dimension. However, the question of which dimension is appropriate is equally difficult, but still quite important from a biological perspective, as the set of neurons may be associated to a space of stimulus with a particular dimension. In our motivating example of place fields, the stimulus space is 2D. However, other famous examples of neurons with receptive fields include neurons in visual cortex which code for the 1D angle of a stimulus [3], and the more recently discovered example of 3D place fields in bats [4]. The dimension is one of the most basic features of the stimulus space which a code should capture.

We will provide some partial answers to the question of in which dimension a code can be realized with convex receptive fields; recall from Exercise 7.1 that without any assumptions on the receptive fields, all codes can be realized in $\mathbb{R}$. Mathematically, the question of the lowest dimension where a code is convex-realizable requires a different approach than the question of realizability. Many of the constructions which provide realizations for the classes of realizable codes seen in the previous section are in very high dimension, and in particular, may not be minimal [13].

Definition 7.13. The *dimension* of a code $\mathcal{C}$ is the minimum $d \in \mathbb{N}$ such that $\mathcal{C}$ has a convex open realization in $\mathbb{R}^d$. If $\mathcal{C}$ is not realizable in any dimension, we say $d = \infty$.

As a basic example, consider the codes containing the codeword **1**, as described in Theorem 7.6. The proof that all such codes are realizable is constructive, and shows how to realize such codes in two dimensions. Hence, any code containing the codeword **1** has dimension $d = 1$ or 2. Fig. 7.5 shows an example of the construction for a simple code.

Obtaining some basic lower bounds on the dimension of a convex realization can be done using the simplicial complex information in the code via the Nerve lemma. A simplicial complex Δ is said to be *d-representable* if there exists a collection of convex sets $\mathcal{U} = \{U_1, \dots, U_n\}$ in $\mathbb{R}^d$ such that $\Delta = \mathcal{N}(\mathcal{U})$. Thus, for any code $\mathcal{C}$, the dimension of $\mathcal{C}$ is at least as high as the minimal d such that $\Delta(\mathcal{C})$ is d-representable [18].

Some of this basic dimensional information can be extracted very quickly using well-known theorems from convex geometry, such as Helly's theorem.

Theorem 7.7 (Helly's Theorem)**.** *Suppose $U_1, \dots, U_k \subset \mathbb{R}^d$ are convex sets with $d < k$. If the intersection of every $d + 1$ of these sets is nonempty, then the full intersection $\bigcap_{i=1}^{k} U_i$ is nonempty.*

Since monomial relationships capture intersection information, we can obtain dimension bounds from the canonical form via Helly's theorem.

Exercise 7.44. Apply Helly's theorem to prove that if $\mathcal{C}$ is a code and $CF(J_{\mathcal{C}})$ contains a monomial of degree $\geq d$, then $\mathcal{C}$ cannot be realized in any dimension less than d.

All of the results outlined here rely on information about $\Delta(\mathcal{C})$ to obtain dimension bounds. However, when considering dimension, the simplicial complex does not tell the whole story. In Exercises 7.8 and 7.37, we saw three codes $\mathcal{C}_1, \mathcal{C}_2$, and $\mathcal{C}_3$, all with the same simplicial complex, where $d(\mathcal{C}_1) = 1$, $d(\mathcal{C}_2) = 2$, and $d(\mathcal{C}_3) = \infty$. Using specifics of the canonical form beyond the monomial/simplicial complex information to compute dimension is still a very open problem.

7.5 CONCLUDING REMARKS

We conclude by giving a description of the ongoing work in this area, highlighting those areas where little is known, in the hopes that the reader will consider taking on these challenges.

In the previous section, we saw partial answers to questions about when a code has a convex realization, and in what dimension such a realization might be possible. In each of these situations, we used the algebraic structure of the neural ideal to recognize the RF relationships which informed us how to apply the result. Further work extracting algebraic signatures of convexity continues [18]. However, many related questions about convexity and dimension remain unanswered. Even for known results, it is not immediately clear how to apply the algebraic characterization, as in the following example.

A code $\mathcal{C}$ is *max intersection-complete* if for any collection of facets (maximal sets) $\rho_i \in \Delta(\mathcal{C})$, we have $\bigcap_{i=1}^{k} \rho_i \in \mathcal{C}$.

Theorem 7.8 (Cruz et al. [17])**.** *If $\mathcal{C}$ is max intersection-complete, then $\mathcal{C}$ is convex.*

Max intersection-complete is a weakening of the condition that $\mathcal{C}$ be intersection-complete. However, unlike intersection-complete codes, an algebraic signature of max intersection-complete codes in $J_{\mathcal{C}}$ is not known.

Similarly, although nonlocal obstructions to convexity have been found (such as those in [16]), it is not known how to detect those obstructions algebraically. The search for other types of obstructions to (or guarantees of) convexity, whether detectable through the neural ideal or not, is still ongoing.

The question of the dimension of a code is likewise still very much open. We have shown that the combinatorial information encoded in the simplicial complex can be used to find a weak upper bound on the dimension, and that Helly's theorem can give us a weak lower bound. Results related to Helly's theorem can be used to find other lower bounds, and in the very specific case of a code containing the all-1s word, we can bound the dimension by 2 [13]. All of this information is based on the information about $\Delta(\mathcal{C})$, however. While this means it can be detected algebraically from the monomial relations, we have also seen that this information by no means sufficient to characterize dimension. In recent work, Zhang and Rosen have characterized the codes of dimension 1 [19], but no such characterization is known for higher dimensions, and no algebraic signature of this characterization is known.

Even in those cases where the code is known to have a convex realization in a low dimension, it is not always clear how to actually construct a realization. In specific cases, a realization can be constructed using the theory of Euler diagrams [20], but the more general question of how to algorithmically construct a convex realization is still open.

Finally, while the results highlighted in this chapter have focused on using the neural ideal under the assumption of convexity, we could also consider loosening this assumption. For example, some recent work has considered the question of dimension of codes under the assumption of connectedness [21]. The machinery of the neural ideal and canonical form is not specific to convexity, and can be used in any context where RF relationships provide useful information.

ACKNOWLEDGMENTS

The authors thank Matthew Macaulay for many helpful suggestions. Carina Curto was supported by NSF DMS-1225666/1537228, NSF DMS-1516881, and an Alfred P. Sloan Research Fellowship. Alan Veliz-Cuba was supported by the Simons Foundation (award ID 516088) and the Mathematical Biosciences Institute (funding through NSF DMS-1440386). Nora Youngs was supported by the Luce Foundation through the Clare Boothe Luce Program.

REFERENCES

[1] J. O'Keefe, J. Dostrovsky, The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat, Brain Res. 34 (1) (1971) 171–175.

[2] C. Curto, V. Itskov, Cell groups reveal structure of stimulus space, PLoS Comput. Biol. 4 (10) (2008) e1000205.

[3] D.H. Hubel, T.N. Wiesel, Place fields of single neurons in the cat's striate cortex, J. Physiol. 148 (3) (1959) 574–591.
[4] M. Yartsev, N. Ulanovsky, Representation of three-dimensional space in the hippocampus of flying bats, Science 340 (6130) (2013) 367–372.
[5] D.A. Cox, J. Little, D. O'Shea, Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra, in: Undergraduate Texts in Mathematics, fourth ed., Springer-Verlag, Berlin, 2015.
[6] D.S. Dummit, R.M. Foote, Abstract Algebra, third ed., John Wiley & Sons, New York, NY, 2004.
[7] E. Miller, B. Sturmfels, Combinatorial commutative algebra, in: Graduate Texts in Mathematics, vol. 227, Springer-Verlag, New York, NY, 2005.
[8] R. Robeva, M.E. Macauley, Algebraic and Combinatorial Computational Biology, Elsevier, Amsterdam, 2018.
[9] C. Curto, V. Itskov, A. Veliz-Cuba, N. Youngs, The neural ring: an algebraic tool for analyzing the intrinsic structure of neural codes, Bull. Math. Biol. 75 (9) (2013) 1571–1611.
[10] L.D. Garica Puente, R. Garcia, R. Kruse, D. Miyata, E. Petersen, N. Youngs, Neural ideals in SageMath, In: Davenport, J.H., Kauers, M., Labahn, G., Urban, J. (eds) Mathematical Software – ICMS 2018. ICMS 2018. Lecture Notes in Computer Science, vol 10931, Springer, 2018.
[11] SageMath, The Sage Mathematics Software System (Version 8.2), Available from: http://www.sagemath.org. (Accessed July 12, 2018).
[12] Ohio Supercomputer Center, Columbus, OH, 1987, Available from: http://osc.edu/ark:/19495/f5s1ph73. (Accessed July 10, 2018).
[13] C. Curto, E. Gross, J. Jeffries, K. Morrison, M. Omar, Z. Rosen, A. Shiu, N. Youngs, What makes a neural code convex?, SIAM J. Appl. Algebr. Geom. 1 (1) (2017) 222–238.
[14] C. Giusti, V. Itskov, A NO-GO theorem for one-layer feedforward networks, Neural Comput. 26 (11) (2014) 2527–2540.
[15] A. Hatcher, Algebraic Topology, Cambridge University Press, Cambridge, UK, 2002.
[16] C. Lienkaemper, A. Shiu, Z. Woodstock, Obstructions to convexity in neural codes, Adv. Appl. Math. 85 (2017) 31–59.
[17] J. Cruz, C. Giusti, V. Itskov, W. Kronholm, On open and closed convex codes, arxiv.org/abs/1609.03502.
[18] C. Curto, E. Gross, J. Jeffries, K. Morrison, Z. Rosen, A. Shiu, N. Youngs, Algebraic signatures of convex and non-convex codes, arxiv.org/abs/1807.02741.
[19] Z. Rosen, Y.X. Zhang, Convex neural codes in dimension 1, arxiv.org/abs/1702.06907.
[20] E. Gross, N. Kazi Obatake, N. Youngs, Neural ideals and stimulus space visualization, Adv. Appl. Math. 95 (2018) 65–95.
[21] R. Mulas, N.M. Tran, Minimal embedding dimensions of connected neural codes, arxiv.org/abs/1706.03999.

FURTHER READING

[22] C. Curto, What can topology tells us about the neural code?, Bull. AMS 54 (1) (2017) 63–78.
[23] W.A. Stein, et al., Sage Mathematics Software (Version 8.2), 2016, Available from: http://www.sagemath.org. (Accessed July 12, 2018).