SYDE 556/750
Simulating Neurobiological Systems
Lecture 10: Symbols and Symbol-like Representations

Andreas Stöckel and Chris Eliasmith

Based on lecture notes by
Chris Eliasmith and Terrence C. Stewart

Nov 10 & 11, 2022

Accompanying Readings: Chapter 3, 4 and 9 of "How to Build a Brain"

Contents

1 Introduction

Note: So far we have mostly been concerned with representing vectors $\mathbf{x}$ in populations of neurons. In this lecture we instead think about the representation of *symbols*, leading up to the Semantic Pointer Architecture (SPA) which we will discuss in the next lecture.

As discussed at the beginning of the course, our ultimate goal is to build models of human brains, and, by extension, human cognition. So far, it might seem as if we had not made much progress in this direction. What we have discussed up to this point is "merely" allowing us to represent vectors in a population of neurons, to compute transformations of those represented values within the connections between populations, and to build dynamical systems treating the represented values as control-theoretic state variables.

In particular, one hallmark of human cognition is *language*, and it is not clear how to apply the NEF to language, or facts expressed in language. Examples of language-based statements that we may want to encode are sentences such as "the number eight comes after the number nine", "all dogs chase cats", or "Anne knows that Bill thinks that Charlie likes Dave".

We attempt to solve this problem by first having a look at other modelling approaches in Cognitive Neuroscience. In particular, we will have a look at so called Vector Symbolic Algebras (VSAs), which can be easily mapped onto the NEF. In the next lecture we then use VSAs to build a particular model of cognition, the Semantic Pointer Architecture (SPA).

2 Neural Theories of Cognition

Note: For more details on the theories of cognition listed here, have a look at Chapter 9 of "How to Build a Brain" [1].

Traditionally, both cognitive scientists and early computer scientists working on artificial intelligence have constructed theories of cognition that involve processing structured information using symbol-based representational frameworks, such as predicate logic. For example, the statements mentioned above could be written like this (this notation is supposed to resemble predicate logic):

- "The number eight comes after the number nine": $\mathbf{isSucc}(\mathtt{EIGHT}, \mathtt{NINE})$.
- "All dogs chase cats": $\forall x \forall y \big(\mathbf{isDog}(x) \land \mathbf{isCat}(y)\big) \rightarrow \mathbf{doesChase}(x, y)$.
- "Anne knows that Bill thinks that Charlie likes Dave":

$$\mathbf{knows}\Big(\mathtt{ANNE}, \text{“}\mathbf{thinks}\big(\mathtt{BILL}, \text{‘}\mathbf{likes}(\mathtt{CHARLIE}, \mathtt{DAVE})\text{’}\big)\text{”}\Big).$$

Both computer scientists and cognitive scientists have built cognitive models that are roughly based on representations akin to the ones listed above. These "symbolic" approaches are generally quite successful in modelling certain aspects of human cognition and can be made to fit behavioural data. However, they do not answer the question of how these symbols are represented and manipulated within a human brain.

2.1 Jackendoff's Challenges for Cognitive Neuroscience

In his 2002 book "Foundations of Language: Brain, Meaning, Grammar, Evolution" [2], linguist Ray Jackendoff poses four challenges aimed at cognitive neuroscientists who aim to build a neural model of language. These challenges are

- **The Binding Problem.** Suppose you see a red square and a blue circle. How does the concept of "red" get bound with the concept of "square", and how is it kept separate from "blue" and "circle"?
- **The Problem of Two.** Consider the sentence "the little star is besides the big star". How do we keep those two uses of the concept "star" separate? For example, if there was a group of neurons representing the concept of "star", how can this group of neurons both represent the "little star", the "big star", as well as the fact that the two stars are next to each other?
- **The Problem of Variables.** Grammar imposes certain rules on sentences. For example, it is "correct" to say "blue x", if x is a noun, but not "blue y", if y is a verb. Correspondingly, the question is how these rules, which rely on placeholders, or "variables", are represented in the brain.
- **Working Memory versus Long-Term Memory.** We can both *use* sentences (and keep them in working memory; i.e., current neural activities), while also being able to *store* them for very long times (long-term memory; synaptic weights). A neural architecture of language must explain how sentences can be transferred from working memory to long-term memory and back. In other words, we must be able to turn representations from neural activities into synaptic weight changes.

2.2 Solution Attempt 1: Neural Synchrony (Oscillations)

One relatively early theory of how the "binding problem" could be solved is the LISA architecture [3], which has been developed in the early 1990s. This architecture solves the binding problem by proposing to exploit "neural synchrony".

A central idea of this approach is that individual groups of neurons represent concepts. This is a so called *localist* representation; individual spatially colocated groups of neurons correspond to individual concepts. Higher-level concepts can be created by connecting groups of neurons. For example, the neurons representing the symbol "cat" can be connected to neurons representing the concepts "furry", "solitary", and "animal" (cf. fig. 1).

Concepts are bound by neurons representing the concept oscillating at the same frequency. That is, for our example of "red square and blue circle", the neuron populations of these concepts would oscillate synchronously in phase, while separate, currently active concepts oscillate with a different frequency and/or out of phase.

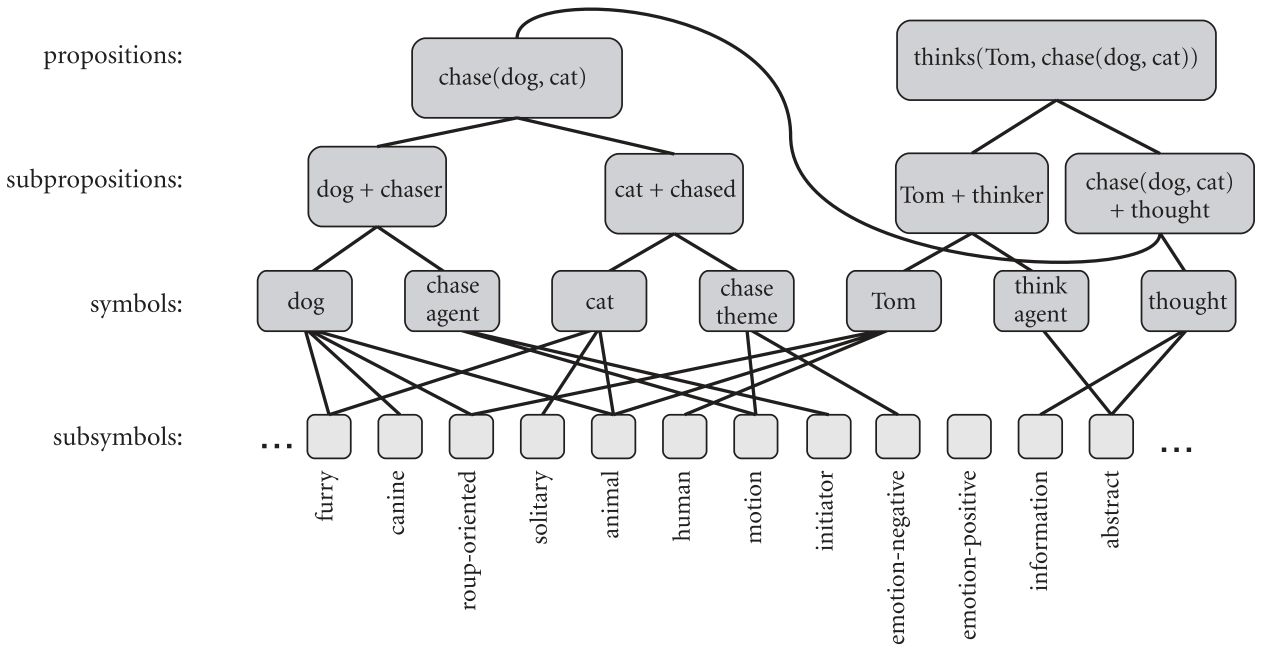

Figure 1: The LISA architecture. Boxes corresponds to groups of neurons representing individual concepts, lines to synaptic conncetions. Concepts are "bound" by neurons oscillating at the same frequency/phase. Figure copied from Figure 9.1 in Eliasmith, 2013 [1], which is in turn adapted from Hummel and Holyoak, 2003 [3].

Let's evaluate this architecture.

- ⊕ The architecture solves the binding problem.
- ● The architecture assumes localist representation, i.e., a few neurons represent a single concept. Most evidence points towards at least a distributed representation.
- ● It is unclear how this architecture would solve the "problem of two" – maybe the same symbol used multiple times could oscillate with two superimposed frequencies.
- ● This architecture cannot solve Jackendoff's third and fourth challenge.
- ⊖ It is unclear how these oscillations are generated and controlled in the first place, i.e., how is language translated into activation of these localist representations?
- ⊖ Furthermore, it us unclear how the representations are processed—which mechanism is reading out the oscillations and decides which neural ensembles are active together?
- ⊖ Finally, we have an exponential explosion of neurons required to represent all kinds of different concepts. For example, assume that there are 10^4 symbols. Then, we have on the order of 10^8 possible second-order combinations of symbols ("subpropositions"), and on the order of 10^{16} possible propositions. This already exceeds the number of neurons in the brain by a factor of one to ten thousand.

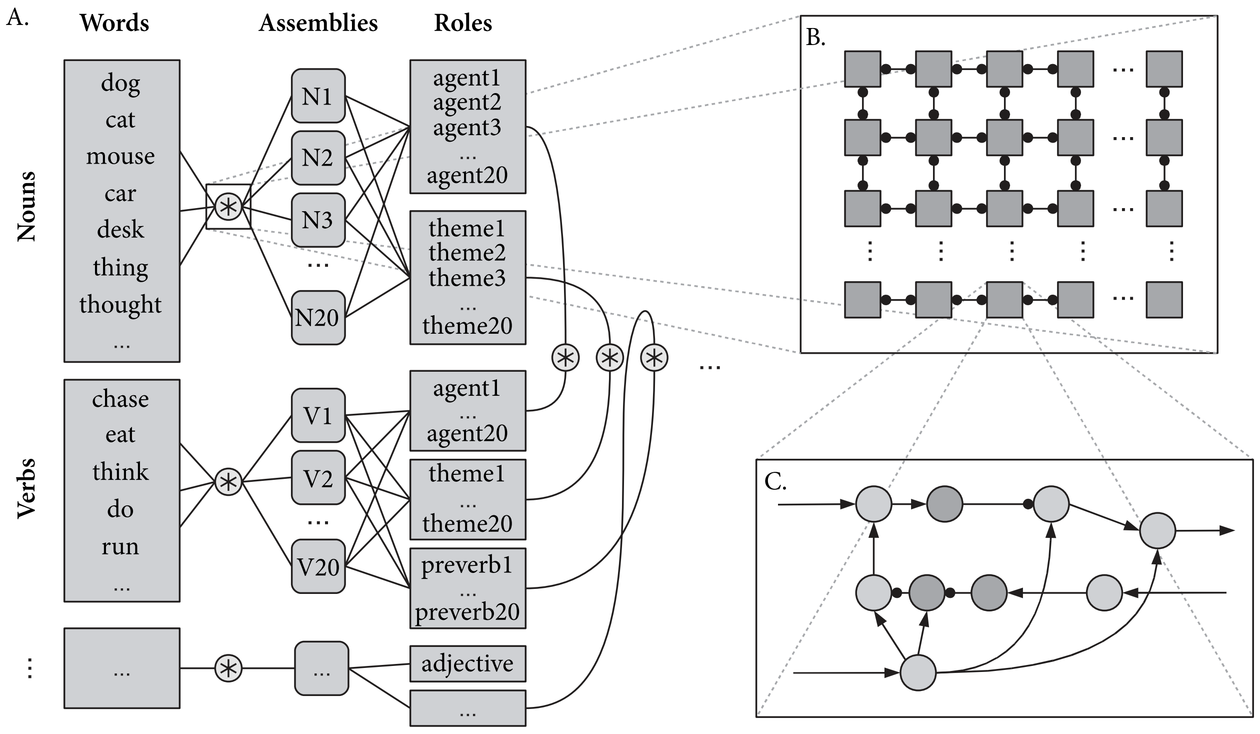

Figure 2: Neural Blackboard Architecture. **(a)** Groups of neurons represent concepts (words)—they are connected via a switchboard circuit (symbolised by ⊛ and depicted in panels **(b)** and **(c)**) to slots corresponding to individual roles in a sentence. These roles are then combined via switchboards into higher-level sentences. Image copied from Figure 9.2 in Eliasmith, 2013 [1], which is in term adapted from van der Velde and de Kamps, 2006 [4].

2.3 Solution Attempt 2: Neural Blackboard Architecture

An attempt at solving all four problems posed by Jackendoff is the "Neural Blackboard Architecture" by van der Velde and de Kamps [4]. The idea is to solve the exponential growth problem posed by synchrony approaches by introducing switchboard circuits that can arbitrarily route signals from neurons representing individual concepts to so called "assemblies" representing bound concepts. These high level concepts can then be associated with specific roles a concept might have in a sentence.

- ⊕ This approach has a much lower resource consumption than LISA.
- ⊕ Can solve all four of Jackendoffs challenges (according to the authors).
- ⊕ Explains limitations of human sentence representation.
- ● This is still a (at least partially) localist representation.
- ⊖ Uses a very particular structure that does not really seem to match biology.
- ⊖ Uses a very large number of neurons; about 500×10^6 to represent simple sentences (those constructed out of 2,000 verbs and 4,000 nouns). For average adult vocabulary

sizes of about 60,000 words, this approach requires more neurons than are in all language areas combined.

- ⊖ Only considers sentence representation, but not how they can be transformed and manipulated.

2.4 Solution Attempt 3: Vector Operators

As mentioned, both approaches discussed above are *localist*. We could instead try to decouple concepts from the underlying neural substrate. That is, populations of neurons can represent different symbols, and "send" represented symbols for processing to other groups of neurons.

This would be more in line with the kind of information processing suggested by the NEF. Knowing that we can represent vectors in neural populations, we could represent symbols as vectors $\mathbf{x} \in \mathbb{R}^d$. For now, let's assume that these vectors $\mathbf{x}$ are randomly generated.

This idea is actually quite old, and there have been multiple suggestions as for how to symbol vectors $\mathbf{x}$ and $\mathbf{y}$ could be bound together. One approach suggested by Smolensky in 1990 [5] is to simply use a tensor product $\mathbf{x} \otimes \mathbf{y}$, a generalisation of a vector outer product to matrices:

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \otimes \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 \\ a_2 b_1 & a_2 b_2 & a_2 b_3 \\ a_3 b_1 & a_3 b_2 & a_3 b_3 \end{pmatrix} \qquad \text{(Outer product)}$$

$$\begin{aligned} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \otimes \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} &= \begin{pmatrix} a_{11} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} & a_{12} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \\ a_{21} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} & a_{22} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \end{pmatrix} \qquad \text{(Tensor product)} \\ &= \begin{pmatrix} a_{11} b_{11} & a_{11} b_{12} & a_{12} b_{11} & a_{12} b_{12} \\ a_{11} b_{21} & a_{11} b_{22} & a_{12} b_{21} & a_{12} b_{22} \\ a_{21} b_{11} & a_{21} b_{12} & a_{22} b_{11} & a_{22} b_{12} \\ a_{21} b_{21} & a_{21} b_{22} & a_{22} b_{21} & a_{22} b_{22} \end{pmatrix} \end{aligned}$$

This way, two vectors $\mathbf{x}$, $\mathbf{y}$ can be bound without losing any information. In contrast to just stacking the two vectors (which would also not incur any information loss), the tensor product has some nice mathematical properties, and can, as outlined by Smolensky in his paper, to a degree be easily implemented in a neural substrate.

- ⊕ Solves the binding problem and the problem of two.
- ● Unclear how to solve Jackendoff's fourth challenge.
- ⊖ The method scales extremely poorly. Every time two vectors are bound, the dimensionality of the resulting structure is squared; we need d^2 dimensions when binding two d-dimensional vectors. In general, for n binding operations we need d^n dimensions.

Note: *Symbolic Architectures and Neuroscience.* All methods discussed so far are trying very hard to map purely symbolic architectures onto a neural substrate. In a sense, neural aspects are treated as *mere implementation details*. This is an instance of the top-down approach we discussed at beginning of the course: mapping high-level cognitive architectures onto biology. In a sense, the hope is that, if successful, neurons would not matter. This is (unfortunately) an assumption many cognitive scientists make.

3 Vector Symbolic Algebras

The last idea—using vectors $\mathbf{x}$ to represent symbols and to then use an operator such as $\otimes$ to bind them—seems to be reasonable, and fits the Neural Engineering Framework quite well. However, what we would like to have is an operator that maintains the dimensionality of the vectors (i.e., takes two vectors of dimensionality d as an input and outputs a vector of dimensionality d), while still allowing us to reconstruct information about the operands.

Example: *Using "+" as a binding operator*. Let's reconsider the "Binding Problem" as originally posed above. We would like to represent the fact that we are seeing a blue square and a red circle. Hence, we could—for now, randomly—generate four d-dimensional vector representing these four concepts, where d is relatively large, for example $d = 64$.

To keep us from having to come up with letters $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, $\mathbf{w}$, … for each concept and remember the mapping between letter and concept, we will just write the vector corresponding to each concept in capital letters, like so: BLUE, RED, CIRCLE, SQUARE. Keep in mind that this is just notation; each of these words is a (random), d-dimensional vector.

Our first attempt at representing the above concept could be to just sum these vectors

$$\mathbf{x} = \texttt{BLUE} + \texttt{SQUARE} + \texttt{RED} + \texttt{CIRCLE}\,.$$

However, notice that mere addition of symbol vectors does not allow us to distinguish which colour belongs to which object. Correspondingly, "+" is not a good choice as a binding operator, but may be used to represent that two separate concepts are currently active.

Mathematically, what we would like is an operator $\circledast$ with the following properties

$$\begin{aligned}
&\circledast : \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}^d\,, && \text{(preservation of dimensionality)}\\
&\mathbf{x} \approx (\mathbf{x} \circledast \mathbf{y}) \circledast \mathbf{y}^{-1}\,, && \text{(approximately reversible)}\\
&0 \approx \langle \mathbf{x} \circledast \mathbf{y}, \mathbf{x} \rangle\,, 0 \approx \langle \mathbf{x} \circledast \mathbf{y}, \mathbf{y} \rangle\,. && \text{(dissimilar to inputs)}
\end{aligned}$$

As we have discussed in the above example, the last property ensures that two concepts that are bound to each other cannot be confused with the original concepts. This prevents us from using an operator such as "+" as a binding operator, which preserves similarity, as graphically depicted in fig. 3, and as can be easily shown:

$$\langle \mathbf{x} + \mathbf{y}, \mathbf{x} \rangle = \langle \mathbf{x}, \mathbf{x} \rangle + \langle \mathbf{x}, \mathbf{y} \rangle \approx \|\mathbf{x}\|^2 + 0\,,$$

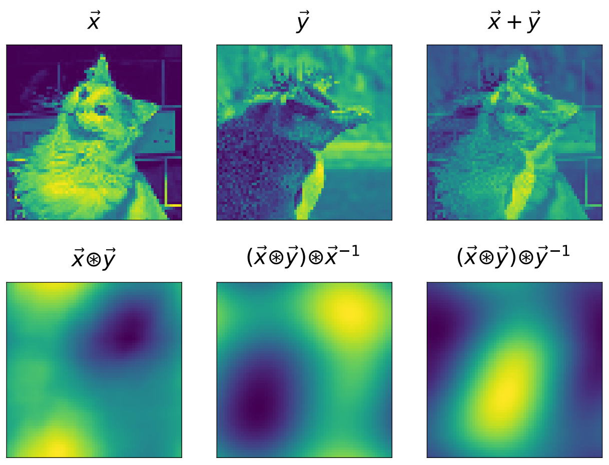
$\vec{x}$
$\vec{y}$
$\vec{x} + \vec{y}$
$\vec{x} \circledast \vec{y}$
$(\vec{x} \circledast \vec{y}) \circledast \vec{x}^{-1}$
$(\vec{x} \circledast \vec{y}) \circledast \vec{y}^{-1}$

Figure 3: Similarity preservation and approximate reversibility of circular convolution. The + operator preserves similarity between **x**, **y** (both the penguin and the cat are still visible), whereas circular convolution $\circledast$ does not (neither the penguin nor the cat are visible). Convolving $\mathbf{x} \circledast \mathbf{y}$ with the pseudo-inverse $\mathbf{x}^{-1}$ and $\mathbf{y}^{-1}$ creates an image that is more similar to **y** and **x**, respectively. The use of images in this example does not imply that we usually apply these methods to uncompressed images. ⌨ *Code*

assuming that **x** and **y** are two high-dimensional random vectors, which are likely to be orthogonal. We will still use "+" to combine multiple concepts into one symbolic vector.

The use of (random) vectors **x** along with a binding operator $\circledast$ to construct cognitive symbolic architectures have been called "Vector Symbolic Architectures" (more properly Algebras; VSAs) by Gayler, 2003 [6]. In the same paper, Gayler argues that such algebras can be used to solve all four challenges put forward by Jackendoff.

Example: *Possible Binding Operators.* Among others, the following binding operators have been proposed for use in vector symbolic algebras:

$$\begin{pmatrix}1\\0\\1\\0\end{pmatrix} \oplus \begin{pmatrix}1\\1\\0\\0\end{pmatrix} = \begin{pmatrix}0\\1\\1\\0\end{pmatrix} \qquad \textit{(XOR)}$$

$$\begin{pmatrix}A\\B\\C\\D\end{pmatrix} \odot \begin{pmatrix}E\\F\\G\\H\end{pmatrix} = \begin{pmatrix}AE\\BF\\CG\\DH\end{pmatrix} \qquad \textit{(Hadamard Product)}$$

$$\begin{pmatrix}A\\B\\C\\D\end{pmatrix} \circledast \begin{pmatrix}E\\F\\G\\H\end{pmatrix} = \begin{pmatrix}AE + BH + CG + DF\\AF + BE + CH + DG\\AG + BF + CE + DH\\AH + BG + CF + DE\end{pmatrix} \qquad \textit{(Circular Convolution)}$$

Circular convolution compresses the outer product by summation along the diagonals:

$$\begin{pmatrix}A\\B\\C\\D\end{pmatrix} \otimes \begin{pmatrix}E\\F\\G\\H\end{pmatrix} = \begin{pmatrix}AE & AF & AG & AH\\BE & BF & BG & BH\\CE & CF & CG & CH\\DE & DF & DG & DH\end{pmatrix} \qquad \textit{(Outer Product)}$$

3.1 Example: Encoding Sentences

We can use the binding operator "⊛" and the plus operator "+" to compress multiple symbols into a single vector. For example, we can now write the above example as

$$\mathbf{x} = \mathtt{BLUE} \circledast \mathtt{SQUARE} + \mathtt{RED} \circledast \mathtt{CIRCLE}\,.$$

We can use the reversibility property to "ask questions". For example, "which object is blue?"

$$\begin{aligned}\mathbf{y} &= \big(\mathtt{BLUE} \circledast \mathtt{SQUARE} + \mathtt{RED} \circledast \mathtt{CIRCLE}\big) \circledast \mathtt{BLUE}^{-1}\\ &= \big(\mathtt{BLUE} \circledast \mathtt{SQUARE}\big) \circledast \mathtt{BLUE}^{-1} + \big(\mathtt{RED} \circledast \mathtt{CIRCLE}\big) \circledast \mathtt{BLUE}^{-1}\\ &\approx \mathtt{SQUARE} + \underbrace{\mathtt{RED} \circledast \mathtt{CIRCLE} \circledast \mathtt{BLUE}^{-1}}_{\text{“noise”}} \approx \mathtt{SQUARE}\,.\end{aligned}$$

Note that we call the term $\mathtt{RED} \circledast \mathtt{CIRCLE} \circledast \mathtt{BLUE}^{-1}$ "noise" because it does not correspond to any meaningful symbol without our vocabulary. Correspondingly, due to the linearity of addition, we know that the vector $\approx \mathtt{SQUARE}$ + "noise" is highly similar to SQUARE, but to no other symbol in our vocabulary—this is what we mean by the last approximate equality in the above equation.

We can use this technique to write down more interesting concepts, such as the sentences we talked about in the previous section:

- "The number eight comes after the number nine":

$$\mathtt{NUMBER} \circledast \mathtt{EIGHT} + \mathtt{SUCC} \circledast \mathtt{NINE}\,.$$

- "The dog chases the cat":

$$\mathtt{DOG} \circledast \mathtt{SUBJ} + \mathtt{CAT} \circledast \mathtt{OBJ} + \mathtt{CHASE} \circledast \mathtt{VERB}\,.$$

- "Anne knows that Bill thinks that Charlie likes Dave":

$$\begin{aligned} \mathtt{SUBJ} \circledast \mathtt{ANNE} + \mathtt{ACT} \circledast \mathtt{KNOWS} + \mathtt{OBJ} \circledast \\ \big(\mathtt{SUBJ} \circledast \mathtt{BILL} + \mathtt{ACT} \circledast \mathtt{THINKS} + \mathtt{OBJ} \circledast \\ \big(\mathtt{SUBJ} \circledast \mathtt{CHARLIE} + \mathtt{ACT} \circledast \mathtt{LIKES} + \mathtt{OBJ} \circledast \mathtt{DAVE}\big)\big)\,. \end{aligned}$$

Note: *Graceful degradation.* The information we can pack into a single vector $\mathbf{x}$ is limited! The larger the number of binding operations, and the number of additions, the lower the precision of operations such as the approximate inverse. While this may sound bad, humans have similar limitations: for example, the nesting depth of sentences, or the number of objects we can keep in working memory is limited. This makes for interesting predictions if we use VSAs to model cognitive phenomena.

3.2 Circular Convolution

In the following, we are going to focus on *circular convolution* as a binding operator. Circular convolution has been used in signal processing for a long time, but has been proposed by Tony Plate in the 1990s as a binding operator [7] and is used extensively in SPA models including Spaun. Circular convolution of two vectors $\mathbf{z} = \mathbf{x} \circledast \mathbf{y}$ is defined as

(Circular Convolution)

$$z_i = \sum_{j=0}^{d-1} x_j y_{i-j \bmod d}\,, \qquad \text{where } \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^d\,, \text{ and } a \bmod b = a - b\left\lfloor \frac{a}{b} \right\rfloor. \tag{1}$$

This operator has the properties we demanded above. A pseudo-inverse $\mathbf{x}^{-1}$ according to the above definition is given as

$$\begin{pmatrix} x_0 & x_1 & x_2 & \ldots & x_{d-2} & x_{d-1} \end{pmatrix}^{-1} = \begin{pmatrix} x_0 & x_{d-1} & x_{d-2} & \ldots & x_2 & x_1 \end{pmatrix}^{-1},$$

i.e., all elements except for the first one are reversed in order.

Note: For most vectors, one can also solve for a more precise inverse by solving a least-squares problem, i.e., minimizing the error $\|\mathbf{y} \circledast \mathbf{x} \circledast \mathbf{x}^{-1} - \mathbf{y}\|^2$. However, the pseudo-inverse as defined above is sufficient in most cases.

One potential problem with circular convolution as defined in eq. (1), especially with respect using it in conjunction with the NEF, is that the operator requires d^2 multiplications of vector coefficients. This means, that, if, for example, $d = 256$, then we need to perform 65536 multiplications. This is a little concerning, particularly because even just multiplying a single pair of scalars with a reasonably small error requires on the order of one hundred pre-neurons. Correspondingly, we would end up with requiring about 6×10^6 neurons just for convolving two semantic pointers.

Luckily, just as "normal" convolution, circular convolution can be expressed as a simple multiplication in the Fourier domain

$$\mathbf{z} = \mathcal{DFT}^{-1}\big(\mathcal{DFT}(\mathbf{x}) \odot \mathcal{DFT}(\mathbf{y})\big),$$

where $\odot$ is element-wise multiplication, or the so-called "Hadamard" product. Recall that the discrete and inverse discrete Fourier transformation are just linear basis transformations, so we can rewrite the above equation as

$$\mathbf{z} = \mathbf{T}^{-1}\big(\mathbf{T}\mathbf{x} \odot \mathbf{T}\mathbf{y}\big),$$

where $\mathbf{T}$ is the constant (complex-valued) matrix describing the Fourier transformation for a d-dimensional vector. As discussed in the lecture about transformations, we can easily compute any linear transformation $\mathbf{T}$ in the connection between two neuron populations by simply multiplying the decoder $\mathbf{D}$ with $\mathbf{T}$. Hence, all we are left with is d complex-values multiplications. Circular convolution is implemented in `nengo.networks.CircularConvolution` as an efficient NEF network.

3.3 Example: Encoding Numbers

Another fun example is encoding integers. Suppose that we have a symbol `ONE` representing the number one. Then, we can define integers as repeated binding of this vector with itself

$$\begin{aligned}
\mathtt{TWO} &= \mathtt{ONE} \circledast \mathtt{ONE}\,, \\
\mathtt{THREE} &= \mathtt{ONE} \circledast \mathtt{TWO} = \mathtt{ONE} \circledast \mathtt{ONE} \circledast \mathtt{ONE}\,, \\
\mathtt{NUMBER}\text{-}k &= \underbrace{\mathtt{ONE} \circledast \mathtt{ONE} \circledast \ldots \circledast \mathtt{ONE}}_{k\text{-times}}\,.
\end{aligned}$$

Using what we know about the relationship between circular convolution and the discrete Fourier transformation, we can also write the above as

$$\mathtt{NUMBER}\text{-}k = \mathcal{DFT}^{-1}\big(\mathcal{DFT}(\mathtt{ONE})^k\big),$$

where exponentiation operator "$\cdot^k$" corresponds to element-wise exponentiation.

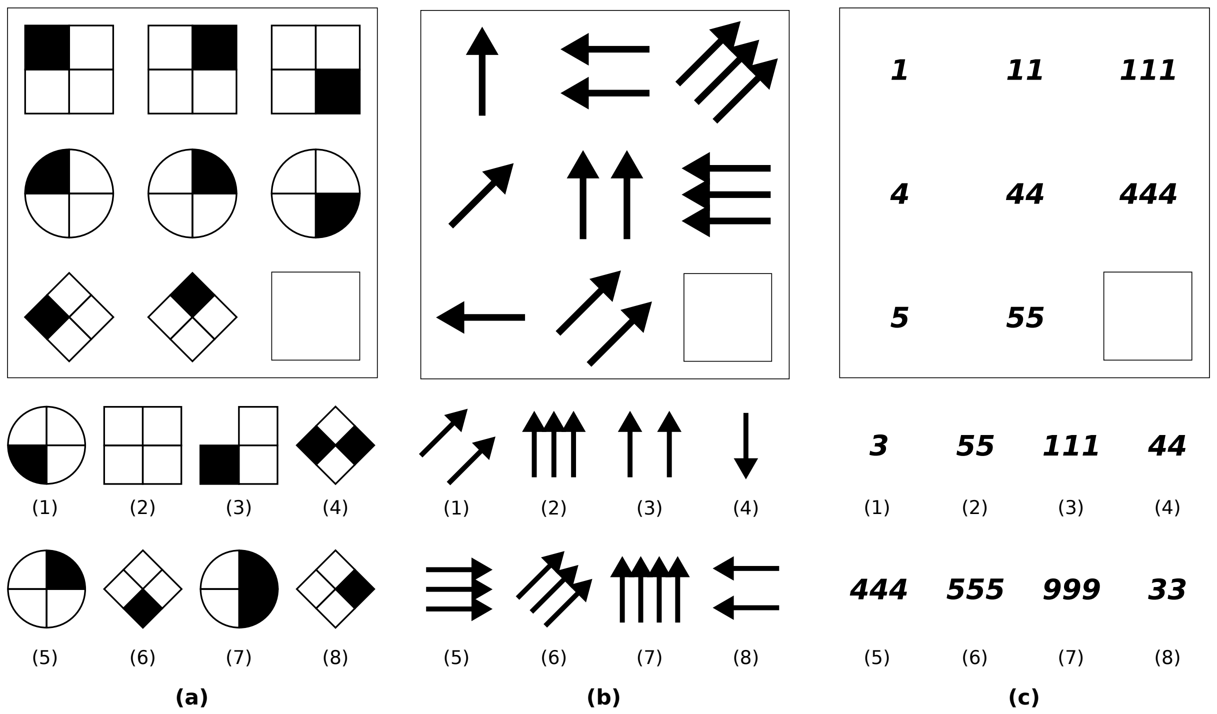

Figure 4: Examples similar to Raven's Progressive Matrices. Participants are asked to select one of eight possible answers that completes the matrix according to the rules underlying its construction.

Note: *Representing continuous values as symbol vectors.* Notice that this allows us to represent any continuous scalar k, and not just integers! We will come back to this when we discuss spatial semantic pointers.

Note: *Unitary vectors.* One potential problem with this approach is that the magnitude of the vector $\mathbf{x}$ might exponentially increase with repeated binding. We thus want binding by ONE to preserve inner products. A symbol vector $\mathbf{x}$ with this property is called a *unitary* vector. "Unitary" because binding with this vector results in "unit" scaling (i.e., no scaling, or scaling by a factor of one). Mathematically, a unitary vector is a vector $\mathbf{x}$ with the following property

$$\|\mathbf{y} \circledast \mathbf{x}\| = \|\mathbf{y}\| \text{ for all } \mathbf{y} \in \mathbb{R}^d .$$

3.4 Example: Raven's Progressive Matrices

Raven's progressive matrices is a commonly used IQ test developed in 1936. Questions on the test typically consist of 3 × 3 matrices filled with symbols arranged according to an unknown, "hidden" rule. One of the cells in the matrix is empty, and participants are asked to complete the matrix according to the rule by choosing one of eight possible cells. Figure 4 shows two examples of what the test could conceptually look like (these are not actual questions on the test; the actual test is kept secret and has not changed since 1936).

Perhaps surprisingly, this IQ test can be solved with a relatively high precision using Vector Symbolic Algebras. To this end, consider the example in fig. 4c and assume that we already have a vision system that translates each cell into a corresponding vector representation. We can argue that this is a problem that we could solve really well using a modern convolutional neural network.

Given this translation between vision and symbol vector, each of the eight given cells could be described in the following way:

$$\begin{aligned}
\mathtt{C1} &= \mathtt{ONE} \circledast \mathtt{P1}, & \mathtt{C5} &= \mathtt{FOUR} \circledast \mathtt{P1} + \mathtt{FOUR} \circledast \mathtt{P2}, \\
\mathtt{C2} &= \mathtt{ONE} \circledast \mathtt{P1} + \mathtt{ONE} \circledast \mathtt{P2}, & \mathtt{C6} &= \mathtt{FOUR} \circledast \mathtt{P1} + \mathtt{FOUR} \circledast \mathtt{P2} + \mathtt{FOUR} \circledast \mathtt{P3}, \\
\mathtt{C3} &= \mathtt{ONE} \circledast \mathtt{P1} + \mathtt{ONE} \circledast \mathtt{P2} + \mathtt{ONE} \circledast \mathtt{P3}, & \mathtt{C7} &= \mathtt{FIVE} \circledast \mathtt{P1}, \\
\mathtt{C4} &= \mathtt{FOUR}, & \mathtt{C8} &= \mathtt{FIVE} \circledast \mathtt{P1} + \mathtt{FIVE} \circledast \mathtt{P2}.
\end{aligned}$$

We can then extract the hidden “rule”, or “transformation” between the cells using the pseudo-inverse (here, we’re only looking at the “horizontal” rule from one cell to the next)

$$\begin{aligned}
\mathtt{T1} &= \mathtt{C2} \circledast \mathtt{C1}^{-1}, & \mathtt{T4} &= \mathtt{C6} \circledast \mathtt{C5}^{-1}, \\
\mathtt{T2} &= \mathtt{C3} \circledast \mathtt{C2}^{-1}, & \mathtt{T5} &= \mathtt{C8} \circledast \mathtt{C7}^{-1}, \\
\mathtt{T3} &= \mathtt{C5} \circledast \mathtt{C4}^{-1}.
\end{aligned}$$

Assuming the rule is consistent, we should be able to extract a clean version of the rule by just averaging these five vectors:

$$\mathtt{T} = \frac{\mathtt{T1} + \mathtt{T2} + \mathtt{T3} + \mathtt{T4} + \mathtt{T5}}{5}.$$

Then, we can make a prediction as for what the ninth cell should contain:

$$\mathtt{C9} = \mathtt{C8} \circledast \mathtt{T} \approx \mathtt{FIVE} \circledast \mathtt{P1} + \mathtt{FIVE} \circledast \mathtt{P2} + \mathtt{FIVE} \circledast \mathtt{P3}.$$

Note: For more detail on the Raven's Progressive Matrices particular example, see [8]. Also, have a look at this video showing Spaun solving this task: 📺 Video. This is based on a memory system showing both recency and primacy effects observed in humans: 📺 Video.

3.5 Revisiting Jackendoff's Challenges

The above example demonstrated that we can solve cognitively challenging tasks using vector symbolic algebras (VSAs). Before we extend VSAs into the semantic pointer architecture (SPA) in the next lecture, let's revisit Jackendoff's challenges, and discuss in how far VSAs are able to solve these.

- **The Binding Problem.** We already discussed this in the example above. In short, if we have concepts such as RED, BLUE, SQUARE, and CIRCLE, we can use the binding operator $\circledast$ to bind concepts, and the addition operator + for concepts that are active at the same time. For example, "red square and blue circle" becomes

$$\mathtt{RED} \circledast \mathtt{SQUARE} + \mathtt{BLUE} \circledast \mathtt{CIRCLE}\,.$$

- **The Problem of Two.** This problem is concerned with the question of how the same concept can be active at the same time in two different contexts. We also had a glance at a solution to this particular problem above: we can simply use symbols denoting the role of individual concepts. Consider the sentence "the little star is besides the big star". We can express this in the following way:

$$\mathtt{OBJ1} \circledast (\mathtt{TYPE} \circledast \mathtt{STAR} + \mathtt{SIZE} \circledast \mathtt{LITTLE}) + \mathtt{OBJ2} \circledast (\mathtt{TYPE} \circledast \mathtt{STAR} + \mathtt{SIZE} \circledast \mathtt{BIG}) + \mathtt{REL} \circledast \mathtt{BESIDES}$$

- **The Problem of Variables.** Here the problem was that language seems to be governed by abstract rules using placeholders. For example, we know that the adjective RED can only be followed by a noun. This rule could be expressed at

$$\mathtt{RULE} = \mathtt{RED} \circledast \mathtt{NOUN}\,.$$

  Then, a variable that can be substituted into the position of "NOUN" is given as

$$\mathtt{VAR} = \mathtt{BALL} \circledast \mathtt{NOUN}^{-1}\,.$$

  Binding the variable with the rule results in the desired concept:

$$\begin{aligned}\mathtt{RULE} \circledast \mathtt{VAR} &= (\mathtt{RED} \circledast \mathtt{NOUN}) \circledast (\mathtt{BALL} \circledast \mathtt{NOUN}^{-1})\\ &= \mathtt{RED} \circledast \mathtt{BALL} \circledast (\mathtt{NOUN} \circledast \mathtt{NOUN}^{-1})\\ &\approx \mathtt{RED} \circledast \mathtt{BALL}\,.\end{aligned}$$

  Note that we did not demand binding operators $\circledast$ to be commutative in general; however, circular convolution happens to be commutative. For the above to work with a non-commutative binding operator, we can define VAR as $\mathtt{NOUN}^{-1} \circledast \mathtt{BALL}$.

- **Working vs. Long-Term Memory.** This problem can be solved quite easily by considering vector symbolic algebras implemented on top of the NEF. Neural activities correspond to concepts that are currently in working memory. Long term memory corresponds to synaptic weights, and hence to functions transforming these concepts—for example, one possible function is an associative long-term memory.

References

[1] Chris Eliasmith. *How to Build a Brain: A Neural Architecture for Biological Cognition*. Oxford Series on Cognitive Models and Architectures. New York, New York: Oxford University Press, 2013. 456 pp. ISBN: 978-0-19-026212-9.

[2] Ray Jackendoff. *Foundations of Language: Brain, Meaning, Grammar, Evolution*. Oxford University Press, USA, 2002. 504 pp. ISBN: 978-0-19-926437-7.

[3] John E. Hummel and Keith J. Holyoak. “A Symbolic-Connectionist Theory of Relational Inference and Generalization.” In: *Psychological Review* 110.2 (2003), pp. 220–264. ISSN: 1939-1471(Electronic),0033-295X(Print). DOI: `10.1037/0033-295X.110.2.220`.

[4] Frank van der Velde and Marc de Kamps. “Neural Blackboard Architectures of Combinatorial Structures in Cognition”. In: *Behavioral and Brain Sciences* 29.1 (2006), pp. 37–70. DOI: `10.1017/S0140525X06009022`.

[5] Paul Smolensky. “Tensor Product Variable Binding and the Representation of Symbolic Structures in Connectionist Systems”. In: *Artificial Intelligence* 46.1 (1990), pp. 159–216. ISSN: 0004-3702. DOI: `https://doi.org/10.1016/0004-3702(90)90007-M`. URL: `http://www.sciencedirect.com/science/article/pii/000437029090007M`.

[6] Ross Gayler. “Vector Symbolic Architectures Answer Jackendoff’s Challenges for Cognitive Neuroscience”. In: *Proceedings of the ICCS/ASCS International Conference on Cognitive Science*. ICCS/ASCS International Conference on Cognitive Science. Sydney, Australia: University of New South Wales, 2003, pp. 133–138.

[7] Tony A. Plate. “Holographic Reduced Representations”. In: *IEEE Transactions on Neural Networks* 6.3 (May 1995), pp. 623–641. ISSN: 1941-0093. DOI: `10.1109/72.377968`.

[8] Daniel Rasmussen and Chris Eliasmith. “A Spiking Neural Model Applied to the Study of Human Performance and Cognitive Decline on Raven’s Advanced Progressive Matrices”. In: *Intelligence* 42 (2014), pp. 53–82. ISSN: 0160-2896. DOI: `https://doi.org/10.1016/j.intell.2013.10.003`. URL: `http://www.sciencedirect.com/science/article/pii/S0160289613001542`.