Supplementary Material

August 17, 2015

Part A - The Neural Engineering Framework

The Neural Engineering Framework (Eliasmith and Anderson, 2003) is a theory that defines methods for implementing mathematical functions in spiking neurons. We use these methods in all of the above simulations to manipulate vectors corresponding to experimental stimuli. To encode a vector into a pattern of neural spikes, the NEF assumes that each neuron has a "preferred" vector for which it fires most strongly. In a given population of neurons, the current J flowing into each neuron i as a function of the value of the represented vector x is given as follows:

$$J_i(x) = \alpha_i(x \cdot e) + J_i^{bias} \tag{1}$$

where α is a gain factor, e is the preferred direction vector or 'encoder', and J^{bias} is the background current entering the neuron. To convert this input current into spiking activity, a neural non-linearity G is applied, yielding an activity rate a for each neuron i:

$$a_i(x) = G_i(J_i(x)) \tag{2}$$

G in this case is defined by the leaky-integrate-and-fire neuron model, which is given by the following equation:

$$G_i(J_i(x)) = \frac{1}{\tau^{ref} - \tau^{RC}\ln(1 - \frac{J^{th}}{J_i(x)})} \tag{3}$$

where τ^{ref} is the neuron's refractory period, τ^{RC} is a decay constant, and J^{th} is the spiking threshold.

Once a vector is encoded into a population of neurons in this manner, it is possible to reconstruct the vector from the spiking activity in the population. This reconstruction is

accomplished by assigning an optimal decoding vector to each neuron, and then using the neural activities in the population to create a weighted combination of these decoding vectors. The following least squares optimization method is used to solve for the decoding vectors d:

$$d = \Gamma^{-1}\Upsilon \qquad \Gamma_{ij} = \sum_x a_i a_j \qquad \Upsilon_j = \sum_x a_j x \tag{4}$$

The sums in the above equations range over a representative sampling of possible values of the vector x. An estimate of x can then be computed as follows:

$$\hat{x} = \sum_i a_i d_i \tag{5}$$

To account for temporal variability in the representation of a vector, the preceding equations are modified slightly to use the spike trains generated from each neural non-linearity rather than the average firing rates. This leads to the following descriptions of the encoding and decoding procedures:

$$a_i(x(t)) = \sum_n \delta(t - t_n) \tag{6}$$

$$\hat{x} = \sum_{i,n} \delta(t - t_{in}) * h(t) d_i = \sum_{i,n} h(t - t_{in}) d_i \tag{7}$$

The first equation treats the activity of each neuron as a collection of spikes over time, while the second equation reconstructs the vector x by weighting each decoding vector by a spike-induced post-synaptic current *h(t)*, and then summing over all of the neurons in the population as before. The equation modeling the post-synaptic current can be varied to account for the presence of different kinds of neurotransmitters in a neural system.

To compute functions on vectors using this framework, the optimization method described in (12) is modified so that the decoders are selected to best reconstruct some function of x, rather than x itself:

$$d^{f(x)} = \Gamma^{-1}\Upsilon^{f(x)} \tag{8}$$

$$\Upsilon_j^{f(x)} = \sum_x a_j f(x) \tag{9}$$

Solving for 'transformational' decoders in this way allows for the creation of neural systems that compute arbitrary non-linear functions. Connections between neural populations are created by providing the decoded output of one neural population as the input to be encoded by another population. As such, the connection weights between two neural populations that compute an arbitrary function can be given as follows:

$$w_{ij} = \alpha_j e_j d_i^{f(x)} \tag{10}$$

We use these methods for solving for all of the connection weights in our model. For further details about the NEF, consult Eliasmith and Anderson (2003).

Part B - Solving for Rule Strengths

To determine the strength of the rules applied in Experiment 3, we use a statistical technique that fits a probability distribution to each rule such that the cumulative effect of applying any set of four rules is to accurately categorize a given stimulus type with probability equal to the experimentally reported proportion of correct responses. For example, if the "consistent" stimuli are categorized accurately 72% of the time, then any random sampling from the distributions we generate will, 72% of the time, result in a set of four rules that function to accurately categorize a consistent stimulus.

To start, we assume that the execution of each rule on a given experimental trial contributes a scalar value to a "coherence score" that measures the extent to which the stimulus under consideration coheres with the knowledge encoded by the rules. Each application of a rule functions to check for the presence of a feature in the stimulus. If the feature is present, the rule boosts the coherence score by an amount equal to its strength. Otherwise the rule has no effect.

Next, we assume that positive categorization judgments occur when the coherence score is over a certain threshold. Based on the empirical data from Lin and Murphy (1997), the threshold must be set such that the coherence scores produced by a given rule set are over the threshold 90% of the time for prototype stimuli, 72% of the time for consistent stimuli, 26% of the time for inconsistent stimuli, and 10% of the time for control stimuli.

To solve for a threshold, we first assume that the coherence scores for the Prototype stimuli are distributed normally with a particular mean and variance. Then, we simply set the threshold so that, 90% of the time, it is below a given coherence score sampled from the distribution. Once this is done, it is straightforward to derive a coherence distribution for each of the other stimuli types by shifting the mean of the first distribution. The result of this process is a set of four coherence score distributions parameterized by the means μ_1, μ_2, μ_3, μ_4, and a constant standard deviation σ.

When each instance of the model is generated, a set of scores are created by sampling once from each distribution. Then, a set of rule strengths are derived with a set of simple linear equations that expresses the relationships that hold between each coherence score, rule strength, and stimulus structure:

$$\begin{aligned} 1 \times r_1 + 1 \times r_2 + 1 \times r_3 + 1 \times r_4 &= Score_1 \\ 1 \times r_1 + 0 \times r_2 + 1 \times r_3 + 1 \times r_4 &= Score_2 \\ 0 \times r_1 + 1 \times r_2 + 1 \times r_3 + 1 \times r_4 &= Score_3 \\ 0 \times r_1 + 0 \times r_2 + 0 \times r_3 + 1 \times r_4 &= Score_4 \end{aligned} \tag{11}$$

The ones and zeros here reflect the analytic feature structure of each stimulus type, while the r values correspond to the strength of each rule. In matrix-vector notation, we can solve for the rule strengths using the normal equation for a linear system:

$$r = (X^T X)^{-1} X^T S \tag{12}$$

where X is a binary matrix encoding the four stimuli structures, r is a vector of rule strengths, and S is a vector of coherence scores. The vector r of rule strengths is scaled by 1.5 to offset the effects of integrating neural representations that do not vary discretely in time, which results in an underestimation of the summations described in equation (19). The vector r is then fixed for each simulated participant before being encoded into the connection weights of the model. Note that effective parameters can be generated for any choice of the first mean μ_1 and the constant standard deviation σ. In the results reported in the paper, we initialize the parameters using this method and apply some minor adjustments to the values of μ_1 and the threshold. However, equivalent results can be obtained by simply selecting an appropriate starting value of μ_1 and σ (1.25 and 0.4 respectively). Further info concerning these choices are detailed in the documentation accompanying the code for the model at https://github.com/pblouw/blouw-etal-2015.

References

Eliasmith, C. and Anderson, C. (2003). *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. MIT Press, Cambridge, MA.

Lin, E. and Murphy, G. (1997). Effects of background knowledge on object categorization and part detection. *Journal of Experimental Psychology*, 23(4):1153–1169.