Lecture 2: Representation in populations of neurons

2.1 Engineered vs. Biological Representation

2.1.1 Engineered Repn.

Codes in information theory are defined by:

1. An encoding procedure: $a = f(x)$
2. A decoding procedure: $x = f^{-1}(a)$

N.B.:

- Both a and x have their own 'alphabets', or 'units'.
- This is for an ideal, or 'lossless' code, otherwise $\hat{x} = g(a) \approx f^{-1}(a)$, which is far more common

One example of such a code is Morse Code where a is a set of numbers whose units are dashes and dots, and x is a set of numbers whose units are the Roman alphabet (plus punctuation).

Another very familiar example is the A/D converters found in many consumer electronics. These convert voltage changes into N bit words. More precisely, we can identify the encoder

$$a_i(x) = \begin{cases} 1, & \text{if } x \bmod 2^i > 2^{i-1} \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

and decoder

$$\hat{x} = \sum_{i=1}^{N} a_i(x)\phi_i, \tag{2.2}$$

where

$$\phi_i = 2^{i-1}.$$

In this example, i indexes a bit number, x is the temperature code, and ϕ is the decoding rule, or weight.

N.B.:

- The encoding is highly nonlinear.
- The decoding is linear.
- This is a distributed representation (at the level of bits). Unlike a photograph (at the level of pixels).
- Side note: Consumer A/D often don't use a temperature code. The above is an example of a 'flash' ADC.

2.1.2 Biological repn.

We can define this in a similar way. Here, the encoders are neurons (which are highly nonlinear), and the decoding weights are unknown. From neuroscience we know that an injected current will cause the cell to fire at a given rate. We then have an encoder:

$$a_i(J) = G_i\left[J\right].$$

N.B.:

- This characterizes a cell *independently* of its being in a neural system.
- Plotting this function by varying J outside of a system gives the neurons *response function*.

To relate this current to its role in a behaving neurobiological system, we need to write it as a function of some external signal, x. Again turning to neuroscience, we notice that cells don't spike outside of a system with no input, but they do inside the system. This suggests that there is a 'background' or bias current stimulating the cell. As well, the current to different cells is changed in different ways in response to changing x, depending on the physiological characteristics of the cell. As a result, we can write J above as:

$$J(x) = \alpha x + J^{bias}.$$

Here, αx, is the 'driving' current, α is the gain that helps determine how the neuron's response changes to changes in x and J^{bias} accounts for the background firing rate. Now our encoder is:

$$a_i(x) = G_i\left[\alpha x + J^{bias}\right]$$

which directly relates an external signal, x, to neural reponses, a. We can determine how well the signal is encoded by finding the decoders:

$$\hat{x} = \sum_i a_i(x)\phi_i.$$

N.B.:

- The second function $a_i(x)$ is called the *tuning curve.*
- The function $G_i\left[\cdot\right]$ can be defined in any number of ways. Obviously a definition that matches known response functions is best. Two useful starting choices are for it to be: 1) simple rectification (computationally cheap); 2) a leaky integrate-and-fire (LIF) response curve (more neurally plausible, and still fairly cheap).
- The choice of $G_i\left[\cdot\right]$ will affect what the determined ϕ_i are.

2.1.3 Biological decoders

To find decoders that support a good representation of x, we want the error between x and the estimate, $\hat{x}$, to be small over the range of the signal. To enforce this constraint, we want to minimize:

$$\begin{aligned} E &= \frac{1}{2}\int_{-1}^{1}\left[x - \sum_{i=1}^{N} a_i(x)\phi_i\right]^2 dx \\ &= \left\langle \left[x - \sum_{i=1}^{N} a_i(x)\phi_i\right]^2 \right\rangle_x, \end{aligned} \tag{2.3}$$

with respect to ϕ_i.[1]

To do so, we can take the derivative of (**??**) with respect to ϕ_i as follows:

$$\begin{aligned} \frac{\delta E}{\delta \phi_i} &= -\frac{1}{2}\int_{-1}^{1} 2\left[x - \sum_{j}^{N} a_j(x)\phi_j\right] a_i(x)\, dx \\ &= -\int_{-1}^{1} a_i(x)x\, dx + \int_{-1}^{1}\sum_{j}^{N} a_i(x)a_j(x)\phi_j\, dx. \end{aligned} \tag{2.4}$$

Setting the derivative to zero and finding the minimum over all i gives

$$\int_{-1}^{1} a_i(x)x\, dx = \sum_{j}^{N}\left(\int_{-1}^{1} a_i(x)a_j(x)\, dx\right)\phi_j, \tag{2.5}$$

which can be written using matrix-vector notation as

$$\Upsilon = \mathbf{\Gamma}\boldsymbol{\phi}. \tag{2.6}$$

Solving for $\boldsymbol{\phi}$ gives

$$\boldsymbol{\phi} = \mathbf{\Gamma}^{-1}\Upsilon, \tag{2.7}$$

where

$$\begin{aligned} \Gamma_{ij} &= \langle a_i(x)a_j(x)\rangle_x \\ \Upsilon_i &= \langle x a_i(x)\rangle_x. \end{aligned}$$

Equivalently, we can write

$$\phi_i = \sum_{j}^{N}\Gamma_{ij}^{-1}\Upsilon_j.$$

[1] Most generally, $\langle f(x)\rangle_x = \frac{\int f(x)w(x)\, dx}{\int w(x)\, dx}$ where $w(x)$ is a weighting factor. For instance, if we let $w(x) = \frac{1}{x^2+a^2}$, then the precision of the resulting representation varies as the distance from the origin. This reflects the well-known Weber-Fechner law from psychophysics.

Minimizing the 'mean square error' (MSE) and thus finding the (mean square) optimal ϕ_i ensures that our representation of x encoded with $a_i(x)$ can be well-decoded over the relevant interval.

N.B.:

Unlike engineered representations, biological ones:

- Result in analog quantities (output signals as rates or time between spikes)
- Need to have the decoders determined in order for us to analyze them as repn.
- Are 'more distributed,' meaning: 1) the failure of an arbitrary neuron will affect the repn by about the same amount regardless of the neuron failing (the effect of a failing bit depends on the order of the bit); 2) the neuron encoding is highly redundant (not so for standard binary codes).

Like engineered representations, biological ones:

- Have encoders that are similar but non-identical
- Are distributed
- Have increasing precision with more encoders
- Have nonlinear encoders and linear decoders

2.2 Adding Noise

Neural systems are noisy (which is good because this lets us use information theory). Noise stems from:

1. Axonal jitter
2. Neurotransmitter vesicle release failures
3. Different amount of transmitter in each vesicle
4. Thermal noise (minor)
5. Ion channel noise (tye number of channels open or closed fluctuates)
6. Network effects.

See also http://diwww.epfl.ch/~gerstner/SPNM/node33.html

Nevertheless, there is good evidence that neurons can pass signals of interest very well (i.e., quite robustly and reproducibly).

N.B.

- Noise is engineered not to be a concern for digital repn, but is a concern for neural repn.

To include noise, we can introduce a noise term, η_i, which is added to the neuron activity, a_i. This makes our estimate of the signal:

$$\hat{x} = \sum_{i=1}^{N} \left(a_i(x) + \eta_i\right) \phi_i. \tag{2.8}$$

To find the decoding weights, ϕ_i, construct and minimize the mean square error as before, averaging over the expected noise, η, as well

$$\begin{aligned} E &= \frac{1}{2} \left\langle \left[x - \sum_{i=1}^{N} \left(a_i(x) + \eta_i\right) \phi_i \right]^2 \right\rangle_{x,\eta} \\ &= \frac{1}{2} \left\langle \left[x - \left(\sum_{i=1}^{N} a_i(x)\phi_i - \sum_{i=1}^{N} \eta_i \phi_i \right) \right]^2 \right\rangle_{x,\eta}. \end{aligned} \tag{2.9}$$

To perform the minimization, assume that the noise is Gaussian, independent, has a mean of zero, and has the same variance for each neuron. The result is:

$$E = \frac{1}{2} \left\langle \left[x - \sum_{i=1}^{N} a_i(x)\phi_i \right]^2 \right\rangle_x + \sum_{i,j=1}^{N} \phi_i \phi_j \left\langle \eta_i \eta_j \right\rangle_\eta. \tag{2.10}$$

Because the noise is independent on each neuron, the noise averages out *except* when $i = j$. So, the average of the $\eta_i \eta_j$ noise is equal to the variance, σ^2, of the noise on the neurons. Thus, the error with noise becomes

$$E = \frac{1}{2} \left\langle \left[x - \sum_{i=1}^{N} a_i(x)\phi_i \right]^2 \right\rangle_x + \sigma^2 \sum_{i=1}^{N} \phi_i^2. \tag{2.11}$$

So, in matrix form, the only change is

$$\Gamma_{ij} = \left\langle a_i(x) a_j(x) \right\rangle_x + \sigma^2 \delta_{ij}$$

N.B.:

- the first squared term in (2.11) is the *error due to static distortion*
- the second term is the *error due to noise*

Note that determining representation in this way relies heavily on considerations of implementational constraints (e.g., noise, number of neurons, range). This is one central themes of the course.

In this week's exercises, you will be examining the relation between the number of neurons in a representation, the noise, and the precision of the representation for a scalar variable. This highlights the difference between neural representation:

- about 100 neurons are needed for a SNR of 100:1
- only 7 bits are needed for the same SNR

2.3 Horizontal eye position

Sections 2.2.1 and 2.2.3, the system description and design specification in the book outline the basic anatomy and response characteristics of neurons in this area.

N.B.:

- neurons in these areas are very well-modeled by LIF neurons
- be sure not to confuse the reponse curves and tuning curves, since they 'look' the same in this example

2.4 Vector Representation

Vectors are far more commonly represented in the nervous system than scalars. Even the neural integrator cells actually represent vectors (the example above is just simplified). Perhaps the best know representation of vectors in the cortex is in primary motor cortex. Apostolos Georgopoulos and his collaborators provided very good evidence that understanding arm movement commands as vectors encoded in a population of neurons in primary motor cortex provided a powerful characterization of this cortical area.

A standard feature of neurons in this area is that they have 'preferred' directions of arm movement. So, a cell will fire most rapidly if the monkey moves its arm in the cell's preferred direction, and less rapidly as the arm movement diverges from the preferred direction. The result is that the tuning curve of these neurons look vaguely like Gaussians or cosines (see slide).

This notion of a 'preferred' vector is thus important for defining the encoding procedure for vector representation. Note that the preferred scalar in the scalar case was simply 'left' and 'right' or 'on' and 'off' and so could be included in the definition of the gain, α.

So, a vector representation can be defined as an encoding:

$$a_i(\mathbf{x}) = G_i \left[\alpha_i \left\langle \tilde{\boldsymbol{\phi}}_i \mathbf{x} \right\rangle_n + J_i^{bias}\right] \tag{2.12}$$

and decoding,

$$\hat{\mathbf{x}} = \sum_{i=1}^{N} a_i(\mathbf{x}) \boldsymbol{\phi}_i. \tag{2.13}$$

The distribution of the 'preferred' (or better yet, 'encoding') vectors, $\tilde{\boldsymbol{\phi}}_i$, are found by looking at the neural system of interest. In the case of arm movements, these approximately evenly spaced around the unit circle. The magnitude of these vectors is always one, because the gain and other neuron parameters (defining G_i) determine the sensitivity of the neuron to changes in magnituded along this direction.

To find the decoding vectors, $\boldsymbol{\phi}_i$, we can perform the same least squares minimization (with noise, which is still scalar) discussed earlier.

2.5 Semicircular canals

I won't discuss the semicircular canal example in much detail. The main point of this section is to highligh the fact that the theory presented here isn't sufficient to determine how to set up a good simulation:

You must look at the empirical details of the system you are interested in.

The methodology discussed earlier emphasizes this as well. The graph of axis vs. evenly distributed 2D vectors demonstrates this fact quite clearly. The motor cortex seems to be like the former and the semicircular canals like the latter.