Lecture 6: Nonlinear transformations & Function representation

6.1 Nonlinear transformations

- Nonlinearities are common in the nervous system, as we know from examining neuron responses. However, these are generally 'small' nonlinearities.
- 'Big' nonlinearities are essential for useful information processing.
- Evidence of such nonlinearities in single neurons (e.g. locust visual system) and in networks (e.g. gain fields)
- The existence of single cell nonlinearities is not yet considered indisputable (and their precise form is clearly not well understood). Nevertheless, let's consider them first.

6.1.1 Nonlinearities in single cells

- If we simply follow our 'recipe' for constructing networks that perform functions, we get something like this for the representations:

$$a_i(x) = G_i\left[\alpha_i \left\langle \tilde{\boldsymbol{\phi}}_i x \right\rangle + J_i^{bias}\right] \tag{6.1}$$

$$\hat{x} = \sum_i a_i(x)\boldsymbol{\phi}_i^x. \tag{6.2}$$

- And this for the transformation:

$$\begin{aligned} c_k(x\cdot y) &= G_k\left[\alpha_k\tilde{\phi}_k(x\cdot y) + J_k^{bias}\right] \\ &= G_k\left[\alpha_k\tilde{\phi}_k\left(\sum_i a_i(x)\phi_i^x \cdot \sum_j b_j(y)\phi_j^y\right) + J_k^{bias}\right] \\ &= G_k\left[\sum_{ij}\omega_{kij}a_i(x)b_j(y) + J_k^{bias}\right], \end{aligned} \tag{6.3}$$

  where the weights $\omega_{kij} = \alpha_k\tilde{\phi}_k\phi_i^x\phi_j^y$ are needed.

- Here, we have a clear nonlinearity needed in our individual neurons, since their response depends on the product of the response of the incoming neurons.
- One proposed mechanism for implementing this kind of multiplication in neurons is 'coincidence detection' (see slide).

- We can examine this by looking at the PSCs. For mathematical simplicity let the $h(t)$ filters be narrow Gaussians (instead of the standard PSCs we use). This givesFrom section **??**, we begin with

$$a_i(x)b_j(y) = \sum_{n,m} h_i(t - t_{in})h_j(t - t_{jm}).$$

- Using narrow Gaussians:

$$\begin{aligned} a_i(x)b_j(y) &= \sum_{n,m} e^{-(t-t_{in})^2/2\Delta^2} e^{-(t-t_{jm})^2/2\Delta^2} \quad (6.4) \\ &= \sum_{n,m} e^{-[(t-t_{in})^2+(t-t_{jm})^2]/2\Delta^2}. \quad (6.5) \end{aligned}$$

- Now re-writing the term in the square brackets

$$\left[(t - t_{in})^2 + (t - t_{jm})^2\right] = 2t^2 - 2t\left(t_{in} + t_{jm}\right) + t_{in}^2 + t_{jm}^2,$$

into which we substitute the equivalence

$$t_{in}^2 + t_{jm}^2 = \frac{\left(t_{in} + t_{jm}\right)^2}{2} + \frac{\left(t_{in} - t_{jm}\right)^2}{2},$$

which gives

$$\begin{aligned} &\left[(t - t_{in})^2 + (t - t_{jm})^2\right] \\ &\quad = 2t^2 - 2t\left(t_{in} + t_{jm}\right) + \frac{\left(t_{in} + t_{jm}\right)^2}{2} + \frac{\left(t_{in} - t_{jm}\right)^2}{2} \\ &\quad = 2\left[t^2 - t\left(t_{in} + t_{jm}\right) + \frac{\left(t_{in} + t_{jm}\right)^2}{4}\right] + \frac{\left(t_{in} - t_{jm}\right)^2}{2} \\ &\quad = 2\left[\left(t - \frac{t_{in} + t_{jm}}{2}\right)^2\right] + \frac{\left(t_{in} - t_{jm}\right)^2}{2}. \end{aligned}$$

- Substituting this back into (6.5) gives the desired result:

$$\begin{aligned} a_i(x)b_j(y) &= \sum_{n,m} e^{-\left[2\left(t-\frac{t_{in}+t_{jm}}{2}\right)^2 + \frac{1}{2}(t_{in}-t_{jm})^2\right]/2\Delta^2} \\ &= \sum_{n,m} e^{-2\left(t-\frac{t_{in}+t_{jm}}{2}\right)^2/2\Delta^2} e^{-\frac{1}{2}(t_{in}-t_{jm})^2/2\Delta^2}. \end{aligned}$$

- This means that we can simply multiply the PSCs currents to do give a good approximation to coincidence detection results.

- However, the big draw back is that every spike in population a must be compared to every spike in population b at the synapses of c. This connectivity is plainly ridiculous. So, although it might be useful in some circuits, coincidence detection (whatever the biophysical details) doesn't seem like a general solution to the problem of multiplying two higher-level signals (e.g, x and y).

6.1.2 Nonlinearities in networks

- Luckily, we can implement nonlinearities at the network level in a way that doesn't rely on biophysically implausible connectivity.
- Nonlinearities in networks are common. Gain fields are one example (see slide) – of course, this could be a result of a better cellular implementation too.
- Again, consider multiplying two variables. This time, however, rather than multiplying the activities of the variables, let us use our definition of representation to some advantage.
- In particular, we form an intermediate representation in a 'middle layer' of neurons (see slide). This will have the dimensionality $D_m = D_x + D_y$.
- We can then extract a transformation for this space by finding optimal decoders for the product $m_1 m_2$. That is, let

$$\hat{f}(\mathbf{m}) = \sum_l d_l(\mathbf{m}) \phi_l^f, \tag{6.6}$$

and solve

$$E_f = \left\langle \left[f(\mathbf{m}) - \hat{f}(\mathbf{m}) \right]^2 \right\rangle_{\mathbf{m}}$$

as usual.

- Now we find the appropriate weights in the network. So substituting the decoding rules into the encoding rules as before. First, the middle layer:

$$\begin{aligned} d_l(\mathbf{m} = [x\, y]) &= G_l \left[\alpha_l \left\langle \tilde{\boldsymbol{\phi}}_l \mathbf{m} \right\rangle + J_l^b \right] \\ &= G_l \left[\alpha_l \left(\tilde{\phi}_l^{m_1} \hat{x} + \tilde{\phi}_l^{m_2} \hat{y} \right) + J_l^b \right] \\ &= G_l \left[\sum_i \omega_{li}^{m_1} a_i(x) + \sum_j \omega_{lj}^{m_2} b_j(y) + J_l^b \right], \end{aligned}$$

where $\omega_{li}^{m_1} = \alpha_l \phi_i^x \tilde{\phi}_l^{m_1}$ and $\omega_{lj}^{m_2} = \alpha_l \phi_j^y \tilde{\phi}_l^{m_2}$.

- Next, to find the weights for the second step, the real transformation, we use the transformation decoders we found in (6.6):

$$\begin{aligned} c_k(f(\mathbf{m})) &= G_k \left[\alpha_k \left(\tilde{\phi}_k f(\mathbf{m}) \right) + J_k^b \right] \\ &= G_k \left[\alpha_k \left(\tilde{\phi}_k \sum_l d_l(\mathbf{m}) \phi_l^f \right) + J_k^b \right] \\ &= G_k \left[\sum_l \omega_{kl} d_l(\mathbf{m}) + J_k^b \right], \end{aligned}$$

where $\omega_{kl} = \alpha_k \tilde{\phi}_k \phi_l^f$.

- As usual, this transformation can be directly implemented in a noisy spiking network (see slides)
- Essentially, we have derived the 'hidden layer' typical in ANNs. It's well known in ANNs that with such a layer you can compute any function of the input (i.e., it's a general function estimator).
- Note also that this suggests a way of understanding nonlinearities in dendrites: we can embed the $\mathbf{m}$ layer into the dendrites. Then some of the nonlinearities would be internal to the cell (but this is physiologically plausible), and we need far fewer cells to do multiplication.

6.2 Negative weights

- I'll skip this, but it's actually quite important to the field, I think.

6.3 Function representation

- We've seen examples of scalar and vector representation (and you've implemented these too; see slides).
- This suggests a 'representational hierarchy' of sorts (ordered by dimensionality).
- Function representation is probably one of the more common kinds of representation because it is so general. If there are reasons to think that the relation between two continuous variables (e.g. light intensity and space, pitch and time, etc.) is represented in part of the brain, then function representation might be most appropriate.
- We'll talk about encoding in lateral intraparietal cortex. Here, there is evidence that objects at the same location cause different firing patterns, so it seems that space and some other variable(s) is (are) related by firing in this area. Let ν be space, and y be the 'other variable'. The we can write

  $$y = x(\nu)$$

  for our representation.
- Since we presume that our representation has a domain of representation, we need to specify a *set* of $x(\nu)$ that can be represented by a neural population.
- We can proceed as we did when thinking about temporal decoding. There we defined a set of functions to get an estimate of the optimal temporal filter over that set. Similarly we want to define a set of functions to get an estimate of the optimal population decoder over that set. So, let's parameterize the set on $\mathbf{A}$ (over some basis):

  $$x(\nu; \mathbf{A}) = \sum_m A_m \Phi_m(\nu) \text{ for } \mathbf{A} \in \rho(\mathbf{A})$$

- Defining $\rho(\mathbf{A})$ is a way of limiting the space spanned by Φ to some subspace of particular interest (just like putting boundaries $[-1, 1]$ on a scalar representation). You can pretend this is a Fourier representation as an example, defining $\rho(\mathbf{A})$ in different ways can get you high, low, or band-pass functions.
- Practically, it's often difficult to define the probability distribution $\rho(\mathbf{A})$. Instead, however, we can take the set of functions that we know to be represented in the system and then determine what vectors, $\mathbf{A}$, can be used to represent these functions. We can then use that set of vectors rather than $\rho(\mathbf{A})$. (Effectively, we are constructing a Monte Carlo estimate of $\rho(\mathbf{A})$).
- The encoding of this space will look familiar:

$$a_i(x(\nu; \mathbf{A})) = a_i(\mathbf{A}) = G_i \left[\alpha_i \left\langle x(\nu; \mathbf{A}) \tilde{\phi}_i(\nu) \right\rangle_\nu + J_i^{bias} \right]$$

- The decoding will too:

$$\hat{x}(\nu; \mathbf{A}) = \sum_i a_i(\mathbf{A}) \phi_i(\nu).$$

- The means of finding the optimal decoders will also be familiar:

$$E = \left\langle \left[x(\nu; \mathbf{A}) - \sum_i \left(a_i(\mathbf{A}) + \eta_i \right) \phi_i(\nu) \right]^2 \right\rangle_{\mathbf{A}, \eta}. \tag{6.7}$$

  Minimizing this error gives, as before,

$$\boldsymbol{\phi}(\nu) = \boldsymbol{\Gamma}^{-1} \boldsymbol{\Upsilon}(\nu), \tag{6.8}$$

  where

$$\Gamma_{ij} = \langle a_i(\mathbf{A}) a_j(\mathbf{A}) \rangle_{\mathbf{A}} + \sigma_\eta^2 \delta_{ij} \tag{6.9}$$

$$\Upsilon_i(\nu) = \langle x(\nu; \mathbf{A}) a_i(\mathbf{A}) \rangle_{\mathbf{A}}. \tag{6.10}$$

- Notes:
  - we write $a_i(\mathbf{A})$ because the ν parameter gets integrated out. This is because it is the $\mathbf{A}$ that really determine which function is being represented by the population
  - this characterization shows that the 'tuning curve' typically measured by neuroscientists will change depending on the input (!) — so it is a warning that we must be very prudent in our choice of 'test' functions for determining the 'real' tuning curve of a cell (we want to choose the appropriate delta function, whatever that is!? — lots of work on neural coding is of this sort. Basically, you show the cell everything you can (white noise), and somehow begin to narrow down the elements of the stimlus set that the cell actually responds to).

– we have encoding functions rather than encoding vectors, but they behave the same way (and if you write them in the $\mathbf{A}$ space, then they are encoding vectors too)

– More specifically, define:

$$\phi_i(\nu) = \sum_m^M q_{im}\Phi_m(\nu). \tag{6.11}$$

and

$$\tilde{\phi}_i(\nu) = \sum_m^M \tilde{q}_{im}\Phi_m(\nu). \tag{6.12}$$

We can now use these as follows:

$$\begin{aligned} a_i(\mathbf{A}) &= G_i\left[\alpha_i\left\langle\sum_{n,m} A_m\Phi_m(\nu)\tilde{q}_{in}\Phi_n(\nu)\right\rangle_\nu + J_i^{bias}\right] \\ &= G_i\left[\alpha_i\left(\sum_{n,m} A_m\tilde{q}_{in}\delta_{nm}\right) + J_i^{bias}\right] \\ &= G_i\left[\alpha_i\left(\sum_m A_m\tilde{q}_{im}\right) + J_i^{bias}\right] \\ &= G_i\left[\alpha_i\left\langle\mathbf{A}\tilde{\mathbf{q}}_i\right\rangle_m + J_i^{bias}\right]. \end{aligned}$$

– We have thus expressed the problem of *function encoding* as one of *vector encoding*, given an orthonormal basis. Similarly, we can express the problem of *function decoding* as one of *vector decoding:*

$$\hat{A}_m = \sum_i a_i(\mathbf{A})q_{im},$$

or

$$\hat{\mathbf{A}} = \sum_i a_i(\mathbf{A})\mathbf{q}_i.$$

– So why bother with function representation? 1. Some neural systems are more naturally thought of this way. 2. It clarifies how treating different vector elements different ways maps to neural processing.

6.4 Representational considerations

- Let's reconsider the formulation we have of what a representation is: this is a philosophy class, after all.
- Representation $\equiv$ {encoding, decoding}.

- This has its own important implications - you don't know a representation until you know how it's used (i.e. decoded). There has been a lot of ink spilled in philosophy regarding that particular issue.
- I think this has the desirable consequence that just because a population may carry information about something (i.e. be effected by it - e.g., bumps to the head) doesn't mean that it *represents* that thing. Representation, then, can only be determined by looking at both input *and* output of a system.
- More specifically, when we look at a particular neural population, what do we know about it's decoders? There are more than one. e.g. $\phi_i^{\mathbf{x}}$ and $\phi_i^{f(\mathbf{x})}$ - how do we know which is the 'representational' decoder? (since $x = f^{-1}(f(x))$)
- Here's what, I think, we can most justifiably stipulate: The represented variable is that variable that all of the other actual decodings can be written as a function of.
- That's a good start, but it doesn't solve the $x = f^{-1}(f(x))$ problem. For this, we have to make an appeal to coherence/consistency: In order for this theory to be consistent with other scientific theories, we *prefer* variables that are part of other theories.
- This would cause us to write things as functions of *velocity* rather than *velocity squared plus two*.