SYDE 556/750
Simulating Neurobiological Systems
Lecture 9: Analysing Representation

Andreas Stöckel and Chris Eliasmith

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

November 4, 2022

Accompanying Readings: Chapter 7 of Neural Engineering

Contents

- **Observation:** Some functions are "harder" to decode than others (larger error)
- **Goal:** Get a better understanding of the types of function that can be decoded
- Tuning curves are a set of basis functions; decoders combine these basis functions

$$\hat{x} = \sum_{i=1}^{n} d_i a_i(x) = \langle \mathbf{d}, \boldsymbol{\alpha}(x) \rangle$$

- Tuning curves are highly similar
- Find basis transformation $\mathbf{T}$ that maximises the information in the basis functions ⇒ PCA

$$\hat{x} = \langle \mathbf{d}, \boldsymbol{\alpha} \rangle = \langle \mathbf{d}\mathbf{T}^{-1}, \mathbf{T}\boldsymbol{\alpha} \rangle$$

- The scale of Eigenvalues corresponding to the individual Principal Compnents is inversely proportional to the noise in the decoding. So if a component has a large Eigenvalue ⇒ this basis function can be decoded well

1 Introduction

Recall that we can write our estimate of $\mathbf{X}$ in matrix notation:

$$\begin{aligned} \hat{\mathbf{X}} &= \mathbf{A}\mathbf{D} \\ &= \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{X}. \end{aligned}$$

where

$$\begin{aligned} \mathbf{D} &= (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{X} \\ &= \gamma^{-1}\Upsilon \end{aligned}$$

or

$$\gamma\mathbf{D} = \Upsilon.$$

We have used packages that take the inverse of $\gamma = \mathbf{A}^T\mathbf{A}$ 'in the right way'. However, because this matrix has no noise term, and because *some* tuning curves are likely to be similar for a large population, that Gram matrix is likely to be singular, or nearly singular so it is *not* invertible. As we have previously seen, there exists a general, and very powerful, technique for analyzing such singular matrices called singular value decomposition (SVD). Recall that SVD decomposition of an $M \times N$ matrix, $\mathbf{B}$, results in three matrices whose product gives $\mathbf{B}$, i.e.,

$$\mathbf{B}_{M\times N} = \mathbf{U}_{M\times N}\mathbf{S}_{N\times N}\mathbf{V}^T_{N\times N}.$$

The matrix $\mathbf{S}$ is a diagonal matrix whose entries are called the *singular values* of $\mathbf{B}$. In the case when $\mathbf{B}$ is square and symmetrical (as with our example), this simplifies to

$$\boldsymbol{\gamma} = \mathbf{U}\mathbf{S}\mathbf{U}^T.$$

2 Singular Value Decomposition of A

In the case where γ is singular (or nearly so), some elements of **S** are zero (or very small), so the inverse of **S** includes infinite (or very large) terms, meaning the inverse of γ is ill-defined (as expected for singular or near singular matrices). In this case, the SVD 'pseudo-inverse' is defined where for $S_i = 0$, the inverse is set to 0. The methods we call for inverting matrices do this 'automatically' (or we add noise to make them invertible). Interestingly, even in the case when a matrix is singular (or nearly so), SVD can be very informative. The columns of **U** whose corresponding singular values are non-zero form an orthonormal basis that spans the range and the corresponding zero elements form an orthonormal basis that spans the null space.

This decomposition is useful for characterizing representation and transformation for a number of reasons.

- First, as already mentioned, the relevant **U** matrix provides an orthogonal basis for both the range and nullity of $\boldsymbol{\gamma}$. Because $\boldsymbol{\gamma}$ tends to be singular, both bases are important.
- Second, when a vector in Υ lies in the range of $\boldsymbol{\gamma}$, the SVD pseudo-inverse guarantees that the corresponding vector from **D** *minimizes* the length of that **D** vector. This is important because given that $\boldsymbol{\gamma}$ is singular, there are an infinite number of solutions for **D**. The solution that provides the shortest vector is a natural and compact choice from the set.
- Third, when a vector in Υ lies in the nullity of $\boldsymbol{\gamma}$, the SVD pseudo-inverse guarantees that the best (in the least squares sense) **D** given Υ will be found. In other words, this 'pseudo-inverse' minimizes the error. Thus, we can use SVD to find the optimal decoding functions, which we can now write as

$$\mathbf{D} = \mathbf{U}\mathbf{S}^{-1}\mathbf{U}^T\mathbf{A}^T\mathbf{X}. \tag{1}$$

  Given the properties of SVD, we know that this is the same solution we would find by constructing the error explicitly (i.e., $E = \left\langle \left[\mathbf{X} - \hat{\mathbf{X}}\right]^2 \right\rangle_{\mathbf{X}}$), taking the derivative, and setting it to zero, as we have previously done.

Recall that regardless of which transformation we need decoders for, we always perform SVD on the same matrix, $\gamma = \mathbf{A}^T\mathbf{A}$. This suggests that understanding the properties of γ can provide general insight into all possible decodings of the population, a_i.

The singular values are useful because they tell us the *importance* of the corresponding **U** vector. There are a number of ways of thinking about 'importance' in this case.

- related to the error that would result if we left a particular vector out of the mapping.
- related to the variance of population firing along the vectors in the γ matrix.
- being the amount of (independent) information about changes in population firing that can be extracted by looking only at data projected onto the corresponding **U** vector.
- In general, we can think of the magnitude of the singular value as telling us how relevant the dimension defined by the corresponding **U** vector is to the identity of the matrix we

have decomposed. Since the matrix we have decomposed is like the correlation matrix of the neuron tuning curves, the large singular values are most important for accounting for the structure of those correlations.

Notice also that the vectors in $\mathbf{U}$ are orthogonal: they provide an (ordered) orthogonal *basis* for that matrix. This is very useful because the original γ matrix was generated by a non-ordered non-orthogonal basis; the neuron tuning curves.

To dive in a bit more, define a point in the 'neuron space' (i.e., the space spanned by the overcomplete neuron tuning curves) as

$$\boldsymbol{\alpha} = a_1\mathbf{e}_1 + a_2\mathbf{e}_2 + \ldots + a_N\mathbf{e}_N.$$

In this notation, the vectors $\mathbf{e}_i$ serve as axes for the state space of the neural population. A point in this space is defined by the neuron firing rates from each neuron in the population (which, taken together, form the vector $\boldsymbol{\alpha}$).

Because the neural responses are non-independently driven by some variable, $\mathbf{x}$, only a *subspace* of the space spanned by the $\mathbf{e}_i$ vectors is ever *actually* occupied by the population.

The γ matrix, because it tells us the correlations between all neurons in the population, provides us with the information we need to determine what that subspace is. When we find the $\mathbf{U}$ vectors in the SVD decomposition, we have characterized that subspace because those are the orthogonal vectors that span it.

Let's see how we can use this to determine what functions can be computed by the particular encoding of $\mathbf{x}$ found in the a_i population:

$$\hat{\mathbf{X}} = \mathbf{A}\mathbf{U}\mathbf{S}^{-1}\mathbf{U}^T\mathbf{A}^T\mathbf{X},$$

or, more simply

$$\hat{\mathbf{X}} = \chi\Phi, \tag{2}$$

where

$$\chi = \mathbf{A}\mathbf{U},$$

and

$$\begin{aligned}\Phi &= \mathbf{S}^{-1}\mathbf{U}^T\mathbf{A}^T\mathbf{X}\\ &= \mathbf{U}^T\mathbf{U}\mathbf{S}^{-1}\mathbf{U}^T\mathbf{A}^T\mathbf{X}\\ &= \mathbf{U}^T\mathbf{D},\end{aligned}$$

Notice that χ and Φ in (2) are rotated versions of $\mathbf{A}$ and $\mathbf{D}$ respectively. Specifically, they are rotated into the coordinate system defined by $\mathbf{U}$. So we can think of $\mathbf{U}$ as the rotation matrix that aligns the first axis of the coordinate system along the dimension with the greatest variance in the encoding of $\mathbf{x}$, the second axis along the dimension with the second greatest variance, and so on. As a result, the χ vectors also end up being orthogonal and ordered by importance. That is, they tell us what can be extracted, and how well it can be extracted, from the neural population. Figure 1 shows an example of how the singular values can be seen as projections of some subspace in the neuron space.

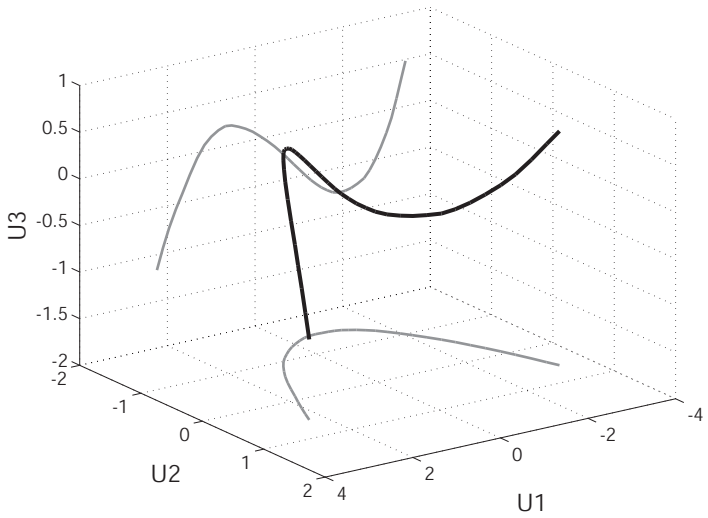

Figure 1: A subspace of neuron activity being projected onto the first few principle component planes. Notice that the axis scales are different, capturing the size of the singular value. Note that U1 is linear in **x**.

We can think of the components of χ as *basis functions* of the space that includes the ensemble of transformations definable on **x** using the encoding in the population a_i.Whichever $\chi(\mathbf{x})$ functions have reasonably large associated singular values, are exactly the functions that we can do a good job of extracting from our encoding of the input space, **x**. Of course, we can also extract any linear combinations of those $\chi(\mathbf{x})$ functions quite well. But, because these functions are ordered, the more useful the 'first' $\chi(\mathbf{x})$ function is for reconstructing some transformation, $f(\mathbf{x})$, the better we can extract that transformation, $f(\mathbf{x})$.

In practice, we can look at the resulting $\chi(\mathbf{x})$ functions and determine what sort of basis we seem to have (see the many examples in the slides). Figure 2 shows an example of doing this for LIF neuron tuning curves in 1D.

3 Function bases and tuning curves

When we do this to standard NEF 1D tuning curves, we approximately find one of the most common polynomial bases used in mathematics, the Legendre basis, $l_i(x)$, which is defined over the interval [-1,1] and results from the orthogonalization of x^n.[1] Scaled versions of the first five elements of this basis are plotted in figure 3.

The similarity between $\chi_m(x)$ and $l_i(x)$ means that this neural population supports the extraction of functions that can be well-estimated using the standard Legendre basis. But the $\chi_m(x)$ functions are ordered by their singular values. Thus, the higher-order polynomial terms are not as well encoded by our population as the lower-order ones. So, computing functions that depend strongly on precise high-order terms will be prone to error.

This is a natural basis to be found from the tuning curves in the linear population. The tuning

[1] One way of expressing the basis is: $l_i(x) = \frac{(-1)^i}{2^i i!} \frac{d^i}{dx^i}\left[\left(1-x^2\right)^i\right]$.

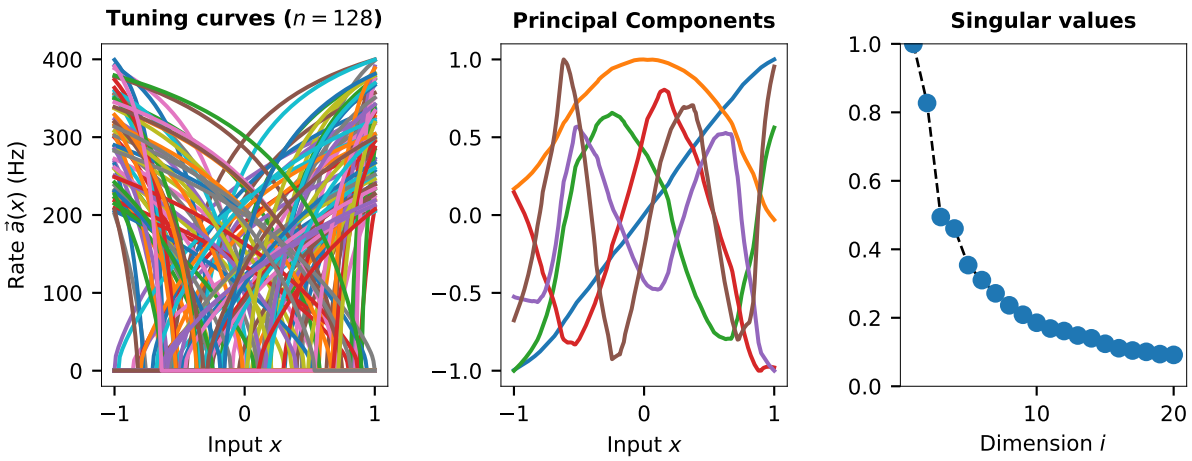

Figure 2: The first few basis functions found from SVD of the activity correlation matrix.

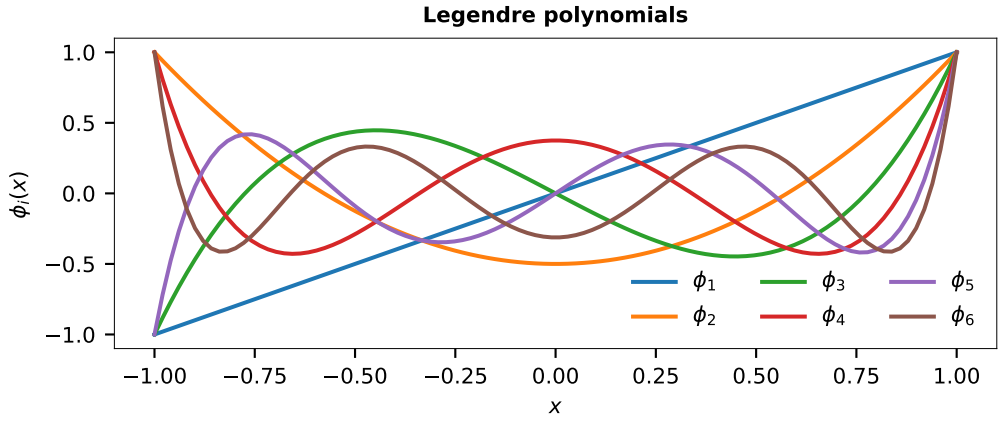

Figure 3: The first few Legendre Polynomials for comparison to those found from SVD of the activity correlation matrix.

curves are very broad, and the polynomial basis is also very broad. These tuning curves are approximately linear, and the more linear basis functions are also the first ones. The Legendre polynomial basis is ordered by decreasing linearity so it should not be too surprising that this population supports the functions in precisely that order.

However, none of this would be true if the population did not do a good job of evenly tiling the input space (see slides for examples). If, for example, there were only high gain neurons, whose slopes were the same sign as their x-intercepts (i.e., if the 'on' and 'off' sub-populations were 'clumped' near x_{max} and x_{min} respectively), we would not expect the linear term to be better supported than the quadratic term. In this sense, the heterogeneity of the population helps it support the more natural ordering of the polynomial basis; clumping would defeat this ordering. Thus, this particular set of $\chi_m(x)$ functions does not just depend on the general 'shape' of the neuron tuning curves, but also on which neurons are included in a population, i.e., the degree of heterogeneity.

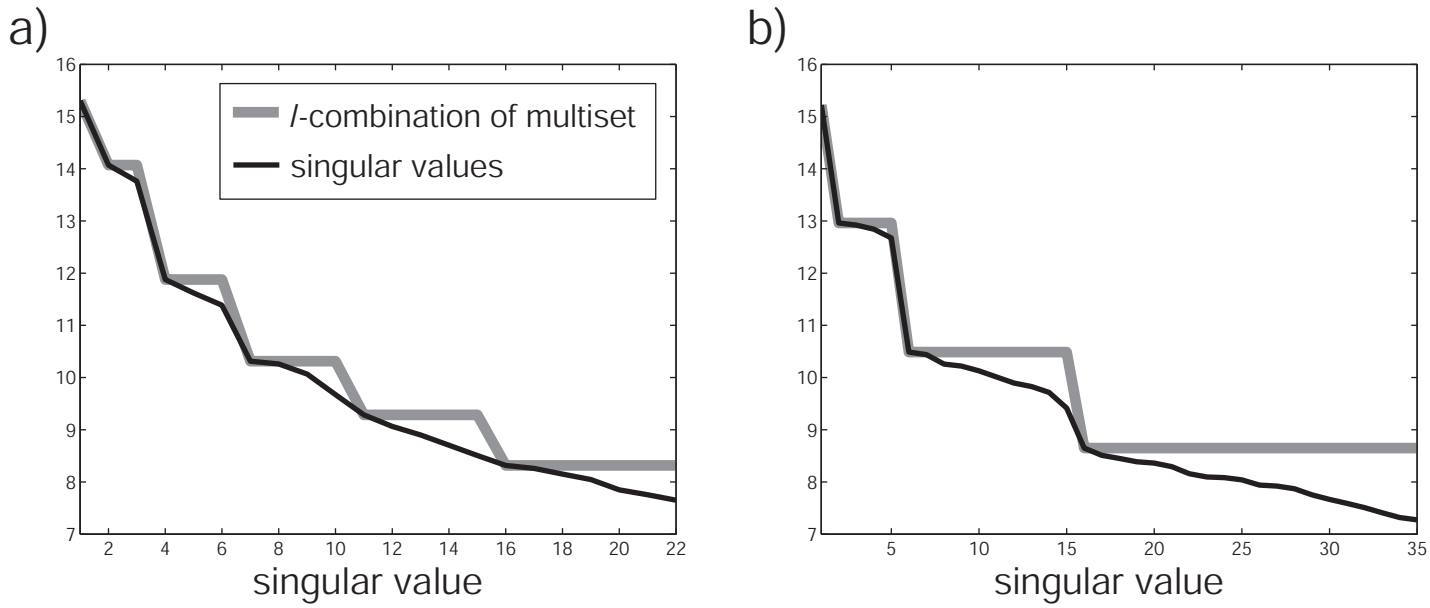

Figure 4: The a) 2D and b) 4D singular values compared to the multiset prediction of how many cross terms there are.

3.1 Singular Values of Vector Representations

There are additional analyses we can perform for vectors of two or more dimensions. As shown in the slides, the same SVD analysis shows that 2D polynomials acts as a useful basis. However, when computing transformations of populations encoding n-dimensional vectors, we must realize that there are additional cross terms (e.g., x_1x_2) that introduce variations in the encoding that are not present in scalar transformations. The expansions are now of the form

$$\begin{aligned} f(\mathbf{x}) &= c_0 + c_1x_1 + c_2x_2 + c_3x_1^2 + c_4x_2^2 + c_5x_1x_2 + \dots \\ &= \sum_{l=0}^{N_{order}} \sum_{n=0}^{l} c_{n,l-n} x_1^n x_2^{l-n}, \end{aligned}$$

where N_{order} is the highest order term in the transformation and l indexes the lth order terms in the expansion (all terms whose exponents *sum* to l are considered lth order terms). As we can see, the cross terms (i.e., $x_1^n x_2^{l-n}$) are quite common. In order to characterize how well an arbitrary function can be decoded from a representation of $\mathbf{x}$, we need to know how big the singular values of these cross terms are as well.

A quick inspection of the singular values of the population reveals that all terms of a given order have singular values of approximately the same magnitude (see Figure 4).

But there is an exponential decay in the magnitude of the singular values as a function of the order of the polynomial. This means that lower-order functions are *significantly* better supported by these kinds of broadly tuned neural populations. That is, the ability to extract higher-order functions drops more quickly with an increase in the order of the function than compared to the scalar case.

In fact, we can determine exactly how many singular values, N_S, there should be for each order, l, in the polynomial for a input space of size D by using the equation for determining l-combinations of a multi-set:

$$N_S(l, D) = \frac{(l + D - 1)!}{l!(D-1)!}. \tag{3}$$

4 Noise

We have been ignoring noise, but most everything stays the same. Rather than the encoding defining a precise subspace (the black line in Figure 1), we can think of it as defining a cloud (e.g. a tube in Figure 1).

However, the noise does not scale with the singular values, it is isometric in the vector space. This means that small singular values are more greatly affected by noise. We can derive this.

We know the error with noise can be written in the vector case as

$$\begin{aligned} E &= \left\langle \left[\mathbf{x} - \sum_i (a_i(\mathbf{x}) + \eta_i)\, \mathbf{d}_i \right]^2 \right\rangle_{\mathbf{x},\eta} \\ &= \left\langle \left[\mathbf{x} - \sum_i a_i(\mathbf{x}) \mathbf{d}_i \right]^2 + \sigma_\eta^2 \sum_i \mathbf{d}_i^2 \right\rangle_{\mathbf{x}}. \end{aligned} \tag{4}$$

We can now use our SVD expressions to give:

$$\begin{aligned} E &= \left\langle \left[\mathbf{x} - \sum_m \chi_m \Phi_m \right]^2 + \sigma_\eta^2 \sum_{i,j} \mathbf{d}_i \delta_{ij} \mathbf{d}_j \right\rangle_{\mathbf{x}} \\ &= \left\langle \left[\mathbf{x} - \sum_m \chi_m \Phi_m \right]^2 + \sigma_\eta^2 \sum_{i,j} \mathbf{d}_i \sum_m U_{im} U_{mj} \mathbf{d}_j \right\rangle_{\mathbf{x}} \\ &= \left\langle \left[\mathbf{x} - \sum_m \chi_m \Phi_m \right]^2 + \sigma_\eta^2 \sum_m \Phi_m^2 \right\rangle_{\mathbf{x}}. \end{aligned} \tag{5}$$

We minimize this error by taking the derivative of (5) and setting it to zero to get an expression for the optimal Φ functions under noise:

$$\begin{aligned} \frac{dE}{d\Phi_n} &= \left\langle 2 \left[\mathbf{x} - \sum_m \chi_m \Phi_m \right] (-\chi_n) + 2\sigma_\eta^2 \Phi_n \right\rangle_{\mathbf{x}} \\ 0 &= -2 \left\langle \chi_n \mathbf{x} \right\rangle_{\mathbf{x}} + 2 \left\langle \sum_m \chi_m \chi_n \Phi_n \right\rangle_{\mathbf{x}} + 2\sigma_\eta^2 \Phi_n \\ \left\langle \chi_n \mathbf{x} \right\rangle_{\mathbf{x}} &= S_n \Phi_n + \sigma_\eta^2 \Phi_n \\ \Phi_n &= \frac{\left\langle \chi_n \mathbf{x} \right\rangle_{\mathbf{x}}}{S_n + \sigma_\eta^2}. \end{aligned} \tag{6}$$

Note that $\sum_m \chi_m \chi_n = S_n$ because the χ are orthogonal, so only the n_{th} term is non-zero. Furthermore,

$$\begin{aligned} \chi^T \chi &= \mathbf{U}^T \mathbf{A}^T \mathbf{A} \mathbf{U} \\ &= \mathbf{U}^T \mathbf{U} \mathbf{S} \mathbf{U}^T \mathbf{U} \\ &= \mathbf{S} \end{aligned}$$

We can use this expression to determine what the residual error will be (i.e., the expected error using these Φ functions). We can now determine the residual error as follows:

$$\begin{aligned} E_r &= \left\langle [\mathbf{x} - \hat{\mathbf{x}}]^2 \right\rangle_{\mathbf{x},\eta} \\ &= \left\langle \left[\mathbf{x} - \sum_m \chi_m \Phi_m \right]^2 \right\rangle_{\mathbf{x},\eta} \\ &= \left\langle \mathbf{x}^2 \right\rangle_{\mathbf{x},\eta} - 2 \left\langle \mathbf{x} \sum_m \chi_m \Phi_m \right\rangle_{\mathbf{x},\eta} + \left\langle \sum_m \chi_m \Phi_m \right\rangle^2_{\mathbf{x},\eta}. \end{aligned}$$

Substituting the expression in (6) for Φ_m gives:

$$\begin{aligned} E_r &= \left\langle \mathbf{x}^2 \right\rangle_{\mathbf{x},\eta} - 2 \left\langle \mathbf{x} \sum_m \chi_m \frac{\langle \chi_m \mathbf{x} \rangle_{\mathbf{x}}}{S_m + \sigma_\eta^2} \right\rangle_{\mathbf{x},\eta} + \left\langle \sum_m \chi_m \frac{\langle \chi_m \mathbf{x} \rangle_{\mathbf{x}}}{S_m + \sigma_\eta^2} \right\rangle^2_{\mathbf{x},\eta} \\ &= \left\langle \mathbf{x}^2 \right\rangle_{\mathbf{x}} - 2 \sum_m \frac{\langle \chi_m \mathbf{x} \rangle^2_{\mathbf{x}}}{S_m + \sigma_\eta^2} + \sum_m (S_m + \sigma_\eta^2) \frac{\langle \chi_m \mathbf{x} \rangle^2_{\mathbf{x}}}{\left(S_m + \sigma_\eta^2\right)^2} \\ &= \left\langle \mathbf{x}^2 \right\rangle_{\mathbf{x}} - \sum_m \frac{\langle \chi_m \mathbf{x} \rangle^2_{\mathbf{x}}}{S_m + \sigma_\eta^2}. \end{aligned}$$

Note that $\left\langle \chi^T \chi \right\rangle_{\eta,m} = S_m + \sigma_\eta^2$, which is derived on page 325 of the course textbook.

We can see this how much the mth basis function, χ_m, reduces the error in our estimate under the influence of noise. Specifically, we know that as the singular value, S_m, approaches the value of the variance of the noise, σ_η^2, the corresponding mth element does not usefully contribute to the representation. This is because the S_m term acts to normalize the effects of the projection onto the non-normalized basis, χ_m.

When the noise becomes near the magnitude of that normalization term (i.e., SNR = 1 or less), the projection onto the relevant χ_m becomes 'mis-normalized' and thus contributes incorrectly to the representation. That is, it will *introduce* error into the representation. This tells us that *those basis functions whose corresponding singular value is equal to or smaller than the noise, should not be used if we want a good representation*.

So, the useful representational space is that space spanned by the basis functions, χ_m, whose corresponding singular values, S_m, are greater than the variance of the noise, σ_η^2, affecting a single neuron. So, we can simply 'lop off' some of the singular values when doing the inverse to get the same result as including a certain amount of noise in our calculation of Γ. This is approximately true, but we haven't done a careful analysis of this relation (between the amount of noise and the number of SVs perserved when inverting γ).

5 Heterogeneity

One thing that becomes clearly important when doing these kinds of analysis is the precise nature of the tuning curves. In the book we show that heterogeneity is a useful balance between usefully tiling a space and ease of construction. Specifically, heterogeneity provides good reduction of error under noise. And, it is easy to construct compared to a perfect, lattice-like spacing of intercepts. (i.e., it's evolutionarily cheap).

We haven't discussed representational capacity above, but both it and 'usefulness' (measured by resistance to noise) are good for heterogeneous populations. This is probably why real neural systems tend to have such tuning curves.