Technical Notes on Linear Regression and Information Theory *

Hiroki Asari†

Septermber 22, 2005

1 Introduction

To understand how the brain processes sensory information, it is important to study the relationship between input stimuli and the output neural responses. Neuroscientists have typically looked at two complementary aspects of neural representations. The first, and best studied, is the encoding process by which a stimulus is converted by the nervous system into neural activity. Less studied is the decoding process, by which experimenters attempt to use neural activity to reconstruct the stimulus that evoked it. To characterize these processes, various methods have been developed to model the stimulus-response functions and to test their performance [2].

Here we briefly overview the basics and logics of these methods. The first part reviews linear regression methods with a certain regularization to find the best linear models. In particular, we will go through how ridge regression is related to the singular value decomposition (for details: [4]). The second part shows how to apply information theory to test the quality of linear filters (for details: [1], [5]). We will discuss the connection of correlation functions to entropy and information, and a way to compute information by exploiting SVD.

2 Linear Regression

A general goal in a regression model is to predict an output y from a vector[1] of inputs $\mathbf{x}$. The linear regression model assumes that the regression function f is linear and has the form

$$\hat{y} = f(\mathbf{x}) = \beta_0 + \sum_j \beta_j x_j = \beta_0 + \mathbf{x}^T \boldsymbol{\beta} \tag{1}$$

where $\hat{y}$ is the estimated output, and $\boldsymbol{\beta}$ (and β_0) are unknown parameters or coefficients. Typically we have a set of n training data: $(y_i, \mathbf{x}_i)$ for $i = 1, \ldots, n$ to estimate the coefficients $\boldsymbol{\beta}$ (and β_0). The most common

*These notes could be rough, and I would appreciate any comments.

†Cold Spring Harbor Laboratory, Watson School of Biological Sciences, One Bungtown Road, Cold Spring Harbor, NY 11724, USA. E-mail: asari@cshl.edu

[1]In this document, we use **boldface** to indicae vectors and matrices

estimation method is to minimize the residual sum of squared errors between the estimated output $\hat{\mathbf{y}}$ and the original output $\mathbf{y}$:

$$E(\boldsymbol{\beta}) = \|\mathbf{y} - \hat{\mathbf{y}}\|^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}),$$

where i-th row of the matrix $\mathbf{X}$ consists of an i-th input vector $\mathbf{x}_i$. For the sake of convenience, here we assume that the outputs have zero mean, $\sum y_i = 0$, that is, $\beta_0 = 0$ in Eq.(1). The least square solution is then given by

$$\hat{\boldsymbol{\beta}}_{\text{ls}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}. \tag{2}$$

Note that $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ is called the pseudoinverse of $\mathbf{X}$, and that $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{y}$ are sometimes referred as auto-correlation and cross-correlation, respectively, in the neurophysiological jargon.

2.1 Ridge regression

In practice, the auto-correlation $\mathbf{X}^T\mathbf{X}$ in Eq.(2) could have some eigenvalues close to zero, leading to an overfitting and a very noisy estimate of the coefficients $\boldsymbol{\beta}$. To address this issue, a regularizer is often introduced to place constraints on the coefficients so that we do not suffer as much from high variability in the estimation [4]. Ridge regression is one of the shrinkage methods to penalize strong deviations of the parameters from zero. That is, the error function to be minimized is

$$E_{\text{ridge}}(\boldsymbol{\beta}, \lambda) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\, \boldsymbol{\beta}^T\boldsymbol{\beta}$$

where the parameter $\lambda \geq 0$ determines the strength of the ridge (power) constraint. The solution for the ridge regression is then given as

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \left(\mathbf{X}^T\mathbf{X} + \lambda\,\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}, \tag{3}$$

where $\mathbf{I}$ is the identity matrix. Note that the solution adds a positive constant to the diagonal of $\mathbf{X}^T\mathbf{X}$ before the inverse, which makes the matrix nonsingular even if $\mathbf{X}^T\mathbf{X}$ is not practically a full-rank matrix.

Singular value decomposition (SVD) The SVD is highly related to the least square solution in Eq.(2) and the ridge regression solution in Eq.(3). The SVD of an $n \times p$ matrix $\mathbf{X}$ has the form

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T, \tag{4}$$

where $\mathbf{U}$ is an $n \times p$ orthonormal matrix whose columns $\mathbf{u}_j$ span the column space of $\mathbf{X}$, and $\mathbf{V}$ is a $p \times p$ orthonormal matrix whose columns span the column space of $\mathbf{X}^T$. $\mathbf{S}$ is a $p \times p$ diagonal matrix of the singular values $s_1 \geq s_2 \geq \cdots \geq s_p \geq 0$. Using SVD, the pseudoinverse of $\mathbf{X}$ can be expressed as

$$\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T = \left(\mathbf{V}\mathbf{S}^2\mathbf{V}^T\right)^{-1}\left(\mathbf{U}\mathbf{S}\mathbf{V}^T\right)^T = \mathbf{V}\mathbf{S}^{-1}\mathbf{U}^T,$$

where $(1/s_1, 1/s_2, \cdots, 1/s_p)$ are on the diagonal of $\mathbf{S}^{-1}$. Therefore, the least square solution in Eq.(2) can be written as

$$\hat{\boldsymbol{\beta}}_{\text{ls}} = \mathbf{V}\mathbf{S}^{-1}\mathbf{U}^T\mathbf{y}.$$

Similarly, the ridge regression solution in Eq.(3) is given as

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \mathbf{V}\left(\mathbf{S}^2 + \lambda\,\mathbf{I}\right)^{-1}\mathbf{S}\mathbf{U}^T\mathbf{y},$$

where the (i,i)-element of the diagonal matrix $\left(\mathbf{S}^2 + \lambda\,\mathbf{I}\right)^{-1}\mathbf{S}$ is $s_i/(s_i^2+\lambda)$.

Now, from Eq.(2), (3), and (4), the estimated outputs $\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$ for the least squares and the ridge regression are written as

$$\begin{aligned}
\hat{\mathbf{y}}_{\text{ls}} &= \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{U}\mathbf{U}^T\mathbf{y}, \\
\hat{\mathbf{y}}_{\text{ridge}} &= \mathbf{X}\left(\mathbf{X}^T\mathbf{X} + \lambda\,\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{U}\mathbf{S}\left(\mathbf{S}^2 + \lambda\,\mathbf{I}\right)^{-1}\mathbf{S}\mathbf{U}^T\mathbf{y} = \sum_{j=1}^{p}\mathbf{u}_j\frac{s_j^2}{s_j^2+\lambda}\mathbf{u}_j^T\mathbf{y},
\end{aligned}$$

respectively. Note that $\mathbf{U}^T\mathbf{y}$ in the least square case are the coordinates of $\mathbf{y}$ with respect to the orthogonal basis $\mathbf{U}$, and the coordinates are shrinked by the factor of $s_i^2/(s_i^2+\lambda)$ in the ridge regression. The estimation noise is then given as

$$\boldsymbol{\eta}_{\text{ls}} = \mathbf{y} - \hat{\mathbf{y}} = \left(\mathbf{I} - \mathbf{U}\mathbf{U}^T\right)\mathbf{y}$$

in the lease squares for example.

3 Information Theory

3.1 Entropy of Gaussian distribution

The probability of the Gaussian distribution for x is given by

$$g(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], \quad \mu = \int g(x)x\,dx, \quad \sigma^2 = \int g(x)x^2\,dx$$

where μ and σ^2 are the mean and the variance of x, respectively. Then it has entropy

$$H(g) = -\int g(x)\log_2 g(x)\,dx = \log_2\sqrt{2\pi e\sigma^2} \quad \text{bit/sample.}$$

Now in general, m-dimensional Gaussian density for $\mathbf{x}$ is

$$G(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^m|\mathbf{A}|}}\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{A}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right],$$

where $\boldsymbol{\mu}$ and $\mathbf{A}$ are the mean and the (symmetric positive semi-definite) covariance matrix, respectively, and $|\mathbf{A}|$ indicates the determinant of $\mathbf{A}$. Then the entropy is

$$H(G) = -\int \cdots \int G(\mathbf{x}) \log_2 G(\mathbf{x})\, d\mathbf{x} = \log_2 \sqrt{(2\pi e)^m\, |\mathbf{A}|} \quad \text{bit/sequence.} \tag{5}$$

When we discuss discrete functions of time, we can think of the correlation function as the analog of the covariance matrix. Therefore, in the case of a single Gaussian signal $x(t)$, we have

$$\mathbf{x} = \begin{pmatrix} x(1) \\ x(2) \\ \vdots \\ x(n) \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} C(0) & C(1) & \cdots & C(n-2) & C(n-1) \\ C(-1) & C(0) & \cdots & C(n-3) & C(n-2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ C(2-n) & C(3-n) & \cdots & C(0) & C(1) \\ C(1-n) & C(2-n) & \cdots & C(-1) & C(0) \end{pmatrix} \tag{6}$$

where $C(\tau)$ is the autocorrelation of $x(t)$:

$$C(\tau) = \lim_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} x(t)\, x(t-\tau).$$

Note that here we have $C(\tau) = C(-\tau)$.

In the case of multiple Gaussian signals $x_i(t)$ for $i = 1, \ldots, m$, we can replace $\mathbf{A}$ in Eq.(5) with the following $m \times m$ block matrix:

$$\mathbf{A} = \begin{pmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} & \cdots & \mathbf{C}_{1m} \\ \mathbf{C}_{21} & \mathbf{C}_{22} & & \vdots \\ \vdots & & \ddots & \vdots \\ \mathbf{C}_{m1} & \cdots & \cdots & \mathbf{C}_{mm} \end{pmatrix} \tag{7}$$

where $\mathbf{C}_{ij}$ is the $n \times n$ cross-correlation matrix[2] between i-th signal $x_i(t)$ and j-th signal $x_j(t)$. Note that $\mathbf{C}_{ij}^T = \mathbf{C}_{ji}$ and thus $\mathbf{A}$ in Eq.(7) is symmetric. Alternatively, we can firstly look at between-set covariances at time τ:

$$\mathbf{C}(\tau) = \begin{pmatrix} C_{11}(\tau) & C_{12}(\tau) & \cdots & C_{1m}(\tau) \\ C_{21}(\tau) & C_{22}(\tau) & & \vdots \\ \vdots & & \ddots & \vdots \\ C_{m1}(\tau) & \cdots & \cdots & C_{mm}(\tau) \end{pmatrix}, \quad \text{where} \quad C_{ij}(\tau) = \lim_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} x_i(t)\, x_j(t-\tau).$$

Then we have the covariance matrix $\mathbf{A}$ as the following $n \times n$ block matrix:

$$\mathbf{A} = \begin{pmatrix} \mathbf{C}(0) & \mathbf{C}(1) & \cdots & \mathbf{C}(n-1) \\ \mathbf{C}(-1) & \mathbf{C}(0) & & \vdots \\ \vdots & & \ddots & \vdots \\ \mathbf{C}(n-1) & \cdots & \cdots & \mathbf{C}(0) \end{pmatrix}.$$

Note the similarity to Eq.(6), and that $\mathbf{A}$ is symmetric since $\mathbf{C}(\tau) = \mathbf{C}(-\tau)$.

[2]The analog of $\mathbf{A}$ in Eq.(6)

3.2 Mutual Information

Entropy measures uncertainty, and information is defined as the difference of entropies, *i.e.*, a reduction of uncertainty [6, 5]. In this way, information theory determines how much information about inputs X is contained in the outputs Y, and can be used to calculate the rates of information transfer. Mutual information between X and Y is defined as

$$I(X,Y) = H(X) - H(X|Y)$$

where the entropy $H(X)$ represents the maximum information that could be encoded in the inputs, and $H(X|Y)$ is the conditional entropy of inputs X given the outputs Y. Alternatively, we can also define $I(X,Y)$ as

$$I(X,Y) = I(Y,X) = H(Y) - H(Y|X)$$

because mutual information is symmetric[3] between X and Y. In the latter expression, the output entropy $H(Y)$ represents the maximal information that could be carried by the system, and $H(Y|X)$ is the entropy in the outputs given the inputs, or the system noise.

Direct method and upper bound estimate of mutual information The direct method calculates information by estimating $H(Y)$ and $H(Y|X)$ from sample data [1]. This is done by separating outputs Y into a deterministic part Y_{det} and a random component by repeating the (same) inputs X many times. Under the additive Gaussian noise assumption for example, Y_{det} can be estimated as the average of Y. Then, we can calculate $I(Y, Y_{\text{det}})$, which gives an estimated upper bound of $I(Y,X)$ if we further assume Y is Gaussian too.

Lower bound estimate of mutual information From data processing inequality theorem, we have $I(Y,X) \geq I(Y,\hat{Y})$ where $\hat{Y}$ is the estimated output of Y from inputs X. If we define $I_G = H(Y) - H(N_G)$, where N_G is the Gaussian process with the same dimension and covariance as the estimated noise $N = Y - \hat{Y}$, then $I(Y,\hat{Y})$ is bounded below by

$$I(Y,\hat{Y}) = H(Y) - H(Y|\hat{Y}) = H(Y) - H(N) \;\geq\; H(Y) - H(N_G) = I_G.$$

The inequality holds because the Gaussian distribution has the maximum entropy given the mean and the covariance. From Eq.(5), an estimate of mutual information is given as

$$I_G = \frac{1}{2}\log_2 \frac{|\mathbf{A}_Y|}{|\mathbf{A}_N|} \tag{8}$$

where $\mathbf{A}_Y$ and $\mathbf{A}_N$ are the covariance matrices of the output Y and the noise N, respectively.

[3] We can also rewrite $I(X,Y) = H(X) + H(Y) - H(X,Y)$ where $H(X,Y)$ is the joint entropy of X and Y.

Computation of information using SVD The most straight forward way to evaluate Eq.(8) is to compute the covariance matrices as in Eq.(6) for a single channel or in Eq.(7) for multiple channels, and to compute their eigenvalues because $|\mathbf{A}| = \prod_i \lambda_i$ where $\lambda_i \in \mathbb{R}$ are the eigenvalues of a symmetric matrix $\mathbf{A}$. However, there are two big difficulties to compute $|\mathbf{A}|$ directly for large datasets: (1) it would be computationally expensive, and (2) it would require huge memory resources. To avoid these issues, we would like to introduce a window to approximate the covariance matrices and exploit SVD (see Eq.(4)) to compute their eigenvalues.

Let us think about a single Gaussian signal $x(t)$ for $t = 1, \ldots, n$, whose covariance matrix $\mathbf{A}$ is given as in Eq.(6). Then the symmetric matrix $\mathbf{A}$ can be written in the following form:

$$\mathbf{A} = \mathbf{X}^T\mathbf{X}, \quad \text{where} \quad \mathbf{X}^T = \frac{1}{\sqrt{n}} \begin{pmatrix} x(1) & x(2) & \cdots & x(n) & & & 0 \\ & x(1) & \cdots & x(n-1) & x(n) & & \\ & & \ddots & \vdots & \vdots & \ddots & \\ 0 & & & x(1) & x(2) & \cdots & x(n) \end{pmatrix} \tag{9}$$

Now, by applying SVD to $\mathbf{X}$, we have the spectral decomposition of $\mathbf{X}^T\mathbf{X}$:

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T, \quad \mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{S}^2\mathbf{V}^T.$$

Here we followed the notations in Eq.(4). Therefore we can easily compute the determinant of the covariance matrix $\mathbf{A}$:

$$|\mathbf{A}| = \left|\mathbf{V}\mathbf{S}^2\mathbf{V}^T\right| = \prod_i s_i^2,$$

where s_i are the singular values of $\mathbf{X}$ and s_i^2 correspond to the eigenvalues of $\mathbf{A}$. In the case of multiple Gaussian signals $x_i(t)$ for $i = 1, \ldots, m$, we can evaluate Eq.(8) in a similar manner. That is, $\mathbf{X}$ that satisfies $\mathbf{A} = \mathbf{X}^T\mathbf{X}$ is given as the following block matrix:

$$\mathbf{X} = \begin{pmatrix} \mathbf{X}_1 & \cdots & \mathbf{X}_m \end{pmatrix} \tag{10}$$

where $\mathbf{X}_i$ is the analog of $\mathbf{X}$ in Eq.(9) for the i-th signal $x_i(t)$.

Although an efficient algorithm for the SVD has been provided elsewhere (*e.g.* svd in Matlab), it might not be a good idea to apply it to $\mathbf{X}$ because of the limit of memory use. In fact, the $[\,(2n-1) \times mn\,]$ matrix $\mathbf{X}$ is bigger than the $[\,mn \times mn\,]$ covariance matrix $\mathbf{A}$ in the single channel case ($m = 1$). Instead, by assuming that there is no correlation between the signals far apart, we can introduce a window of length $k\ (\leq n)$ to approximate the covariance matrix $\mathbf{A}$, *i.e.*,

$$\mathbf{A} \approx \mathbf{A}' = \mathbf{X}'^T\mathbf{X}',$$

where $\mathbf{X}'$ is the $[(n+k-1)\times k]$ matrix corresponding to the upper-left corner[4] of $\mathbf{X}$ in Eq.(9) in the single channel case[5]. Note that having the window length k result in the same approximation level as having the bin size $2\pi k/n$ for the analysis in the Fourier domain (see Appendix).

Furthermore, we can (randomly) pick up $l\,(\leq n-k+1)$ samples to obtain an $\mathbf{X}'$ analog, resulting in the $[l\times k]$ and $[l\times km]$ matrices in the single and multiple channel case, respectively:

$$\mathbf{X}'' = \frac{1}{\sqrt{l}}\begin{pmatrix} x(i_1) & x(i_1+1) & \cdots & x(i_1+k-1) \\ x(i_2) & x(i_2+1) & \cdots & x(i_2+k-1) \\ \vdots & \vdots & & \vdots \\ x(i_l) & x(i_l+1) & \cdots & x(i_l+k-1) \end{pmatrix}$$

In this way, we can reasonably approximate the covariance matrices and evaluate Eq.(8) in the time domain.

A Appendix

A.1 Computation of mutual information in the Fourier domain

Although the Eq.(8) holds in any orthonormal basis, it is in most cases evaluated in the Fourier domain under the assumption of stationary (time translation-invariant) ensembles [5]. The main reason is that the covariance matrices in the Fourier domain are diagonal because the Fourier transform is the expansion using a set of *orthogonal* basis functions. Therefore, different frequency components can be thought of as independent variables, and the power spectrum measures the variances of these independent variables:

$$\log_2|\mathbf{A}_Y| = \sum_\omega \log_2 P_Y(\omega), \quad \log_2|\mathbf{A}_N| = \sum_\omega \log_2 P_N(\omega)$$

where $P_Y(\omega)$ and $P_N(\omega)$ are the power spectral densities of the outputs and the noise, respectively. Note that the power spectral density can be obtained by the squared Fourier coefficients of the signals, or the Fourier transform of the auto-correlation function[6]. Then we have

$$I_\omega = \frac{1}{2}\log_2\frac{P_Y(\omega)}{P_N(\omega)}, \quad I_{LB} = \sum_\omega I_\omega = \sum_{\omega=0}^{f_c}\log_2\frac{P_Y(\omega)}{P_N(\omega)} \quad \text{bits} \tag{11}$$

where f_c is the Nyquist frequency, and I_ω is the information at frequency ω. Note that $I_\omega = I_{2f_c-\omega}$

In the case of multiple dynamic channels, the evaluation of the Eq.(8) in the time domain directly using Eq.(7) takes a while and need huge resources, because we need to consider the correlation between the channels as well. For example, in a linear decoding (reconstruction) model where neural response $r(t)$

[4]The first k columns of $\mathbf{X}$ in Eq.(9) in essence.

[5]In the multiple channel case, by considering the analog of Eq.(10), we have the $[(n+k-1)\times km]$ matrix $\mathbf{X}'$.

[6]This is known as the Wiener-Khintchine theorem, meaning that, for large n in Eq.(6), the eigenvalues of $\mathbf{A}$ correspond to the power spectral density of $\mathbf{x}$.

is used to estimate input spectrogram $S(t, f)$, the covariance matrix of S has $O(n^2 m^2)$ elements where n is the sample size in time and m is the number of frequency bands in S or the number of channels[7]. In contrast, two-dimensional Fourier transform leads to a diagonal covariance matrix of the Fourier coefficients. Therefore, it is much easier to evaluate Eq.(8) in the Fourier domains, and the lower-bound of mutual information $I(S, r)$ is given as

$$I_{LB} = \sum_{n,\,m} I(\omega_n,\, \omega_m) = \sum_{n,\,m} \log_2 \frac{P_S(\omega_n,\, \omega_m)}{P_N(\omega_n,\, \omega_m)} \quad \text{bits},$$

where ω_n and ω_m are the Fourier domains corresponding to time and frequency in the spectrogram, respectively. $I(\omega_n,\, \omega_m)$ is the information at $(\omega_n,\, \omega_m)$, and P_S and P_N are the squared 2-D Fourier coefficients of the input spectrogram and the reconstruction noise, respectively.

A.2 Signal-to-noise ratio and coherence function

Several equivalent formulae for the Eq.(11) are known in the linear (least square) model, using signal-to-noise ratio (SNR) and coherence function [3, 5]. Let $Y(t)$ and $\hat{Y}(t)$ be the output and its estimate from inputs $X(t)$, respectively. Then the estimated noise is given as $N(t) = Y(t) - \hat{Y}(t)$, and the SNR is defined as

$$\text{SNR}(\omega) = \frac{P_{\hat{Y}}(\omega)}{P_N(\omega)} = \frac{P_Y(\omega)}{P_N(\omega)} - 1,$$

where P_Y, $P_{\hat{Y}}$ and P_N are the power spectral densities of Y, $\hat{Y}(t)$ and $N(t)$, respectively. Then, the lower-bound of information can be written as

$$I_{LB} = \sum_{\omega} \log_2 \big[1 + \text{SNR}(\omega)\big] \quad \text{bits}.$$

Note that the SNR can also be defined as

$$\text{SNR}(\omega) = \frac{P_Y(\omega)}{P_{N_{\text{eff}}}(\omega)},$$

where $P_{N_{\text{eff}}}$ is the power spectral density of the effective noise $N_{\text{eff}}(t)$ uncorrelated to the original output $Y(t)$:

$$\hat{Y}(t) = g(t) * \big(Y(t) + N_{\text{eff}}(t)\big).$$

Here $*$ denotes the convolution, and the function $g(t)$ is chosen so that the cross-correlation between $Y(t)$ and $N_{\text{eff}}(t)$ is equal to zero for any t. Then, the Fourier transform of $g(t)$ is called the coherence function $\tilde{g}(\omega)$, and the lower-bound of information can be given by

$$I_{LB} = -\sum_{\omega} \log_2 \big[1 - \tilde{g}(\omega)\big] \quad \text{bits}.$$

[7] To avoid this issue, we introduced the SVD method here.

In the linear model, note that the coherence function between the inputs $X(t)$ and the outputs $Y(t)$ can be rewritten as

$$\tilde{g}(\omega) = \frac{|P_{XY}(\omega)|^2}{P_X(\omega)P_Y(\omega)} = \frac{\text{SNR}(\omega)}{1 + \text{SNR}(\omega)}$$

where P_{XY} is the Fourier transform of cross-correlation between X and Y, and P_X and P_Y are the power spectral densities of X and Y, respectively.

References

[1] Borst, A. and Theunissen, F. (1999). Information theory and neural coding. *Nat Neurosci* **2**(11): 947–957.

[2] Dayan, P. and Abbott, L. (2001). Theoretical Neuroscience Computational and Mathematical Modeling of Neural Systems. Computational NeuroscienceMIT Press.

[3] Gabbiani, F. (1996). Coding of time-varing signals in spike trains of linear and half-wave rectifying neurons. *Network* **7**: 61–85.

[4] Hastie, T., Tibshirani, R., and Friedman, J. (2001). The elements of statistical learning theory. Springer, New York.

[5] Rieke, F., Warland, D., Steveninch, R., and Bialek, W. (1997). Spikes: Exploring the Neural Code. MIT Press, Cambridge, MA.

[6] Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal* **27**: 379–423, 623–656.