Appendix A

Technical Notes

This appendix overviews technical details and algorithms in close relation to the data analyses and simulations in this thesis. All these methods are widely used in various scientific fields such as statistical learning and psychophysics as well as neuroscience (for reviews, see Cover and Thomas, 1991; Hertz et al., 1991; Rieke et al., 1997; Dayan and Abbott, 2001; Duda et al., 2001; Hastie et al., 2001; Nelles, 2001; MacKay, 2003). Section A.1 describes matrix decomposition methods and their characteristics for extracting features in given data sets and for simplifying them in the sense of data compression or dimension reduction under arbitrary criteria of interest. In Section A.2, I explain linear regression and regularization techniques, which are closely related to matrix decomposition methods. Section A.3 then reviews the basics of artificial neural networks (multilayer perceptrons) and support vector regression, in order to provide a unifying view of (nonlinear) data fitting techniques often used in neuroscience. Finally, I give some backgrounds on information theory in Section A.4, and show a little survey on entropy estimation by exploiting file compression methods.

A.1 Matrix Decomposition

Let an $N \times M$ matrix $\mathbf{X}$ be an original data matrix, each column of which contains the N-dimensional data or channels for one of M observations or samples (typically, $M \geq N$). The matrix $\mathbf{X}$ is then decomposed as:

$$\mathbf{X} = \mathbf{AS} + \text{noise}, \tag{A.1}$$

where the dimensions of the factors $\mathbf{A}$ and $\mathbf{S}$ are $N \times R$ and $R \times M$, respectively. The rank of factorization, R, is chosen as: $NR + RM < NM$, to compress the original NM elements in $\mathbf{X}$ into a smaller number of elements, NR in $\mathbf{A}$ plus RM in $\mathbf{S}$, with some "noise" level. In blind source separation (BSS), $\mathbf{A}$ and $\mathbf{S}$ are called mixing and source matrices, respectively, and we can express the solution to Eq.(A.1) using a projection or unmixing matrix $\mathbf{W}$:

$$\mathbf{Y} = \mathbf{WX}, \tag{A.2}$$

where the matrix $\mathbf{Y}$ is the estimate of the source $\mathbf{S}$. For simplicity, we assume without loss of generality that each row of $\mathbf{X}$ has zero mean.

A.1.1 Principal Component Analysis

Principal component analysis (PCA) projects N-dimensional data onto a lower R-dimensional subspace as in Eq.(A.2) in a way that is optimal in a sum-squared error sense. That is, PCA represents an original data set or variables $\mathbf{x} \in \mathbb{R}^N$ by using a new set of variables $\mathbf{y} \in \mathbb{R}^R$, called *principal components*, that captures most of the summed squared vector lengths of the data. All the principal components y_r (for $r = 1, \ldots, R \leq N$) are orthogonal to each other, and the projection coefficients w_{rn} are chosen so as to satisfy the following condition.

> The variance of the first principal component y_1 is the maximum among all possible choices of w_{1n}, and the variance of the r-th principal component y_r (for

$r = 2, \ldots, R$) is the maximum among all possible linear combinations $\sum_n w_{rn} x_n$ that are uncorrelated to $y_{r'}$ (for $r' = 1, \ldots, r-1$).

By "projection coefficients" we mean:

$$\|\mathbf{w}_r\|^2 = \sum_n w_{nr}^2 = 1. \tag{A.3}$$

From Eq.(A.2), the first principal component $\mathbf{y}_1^\top = (y_{11}, \ldots, y_{1M})$ can be written as:

$$\mathbf{y}_1^\top = \mathbf{w}_1^\top \mathbf{X}, \tag{A.4}$$

where $\mathbf{w}_1^\top = (w_{11}, \ldots, w_{1N})$. The mean and the variance of $\mathbf{y}_1$ are given as:

$$\mathbb{E}\big[\mathbf{y}_1\big] \overset{\text{def}}{=} \frac{1}{M}\sum_{m=1}^{M} y_{1m} = \frac{1}{M}\sum_{m=1}^{M}\sum_{n=1}^{N} w_{1n} x_{nm} = \frac{1}{M}\sum_{n=1}^{N} w_{1n}\left(\sum_{m=1}^{M} x_{nm}\right) = 0, \tag{A.5}$$

$$\mathbb{V}\big[\mathbf{y}_1\big] \overset{\text{def}}{=} \frac{1}{M-1}\mathbf{y}_1^\top \mathbf{y}_1 = \frac{1}{M-1}\left(\mathbf{X}^\top \mathbf{w}_1\right)^\top \mathbf{X}^\top \mathbf{w}_1 = \mathbf{w}_1^\top \mathbf{\Lambda} \mathbf{w}_1 \geq 0, \tag{A.6}$$

respectively, where $\mathbf{\Lambda} = \mathbf{X}\mathbf{X}^\top/(M-1)$ is the covariance matrix of $\mathbf{X}$. We can then use the Lagrange multiplier method to find a projection $\mathbf{w}_1$ that maximizes the variance $\mathbb{V}\big[\mathbf{y}_1\big]$ subject to Eq.(A.3):

$$\mathbf{w}_1 = \underset{\|\mathbf{w}\|=1}{\arg\max}\, \mathbb{V}\big[\mathbf{w}^\top \mathbf{X}\big] = \underset{\mathbf{w}}{\arg\max}\, \mathbf{w}^\top \mathbf{\Lambda} \mathbf{w} - \lambda_1 \left(\mathbf{w}^\top \mathbf{w} - 1\right), \tag{A.7}$$

where λ_1 is a Lagrange multiplier. Let J_1 be the objective in Eq.(A.7), and we have:

$$\frac{\partial J_1}{\partial \mathbf{w}_1} = 2\mathbf{\Lambda}\mathbf{w}_1 - 2\lambda_1 \mathbf{w}_1 = \mathbf{0}, \quad \text{that is,} \quad (\mathbf{\Lambda} - \lambda_1 \mathbf{I})\, \mathbf{w}_1 = \mathbf{0}, \tag{A.8}$$

where $\mathbf{I}$ is the identity matrix. Substituting Eq.(A.8) into Eq.(A.6), we have:

$$\mathbb{V}\big[\mathbf{y}_1\big] = \mathbf{w}_1^\top \mathbf{\Lambda} \mathbf{w}_1 = \mathbf{w}_1^\top \lambda_1 \mathbf{w}_1 = \lambda_1, \tag{A.9}$$

where the last equality holds from Eq.(A.3). Therefore, from Eqs.(A.8) and (A.9), λ_1 and $\mathbf{w}_1$ can be given as the maximum eigenvalue of $\mathbf{\Lambda}$ and the corresponding eigenvector, respectively.

Similarly, we can find the projections $\mathbf{w}_r$ for the r-th principal components $\mathbf{y}_r$ by mathematical induction. Let us assume that the projections $\mathbf{w}_i$ for $i = 1, \ldots, r-1$ satisfy the following conditions:

$$(\mathbf{\Lambda} - \lambda_i \mathbf{I})\, \mathbf{w}_i = \mathbf{0}, \tag{A.10}$$

$$\mathbf{w}_i^\top \mathbf{w}_j = \delta_{ij}, \tag{A.11}$$

where δ_{ij} is the Kronecker delta:

$$\delta_{ij} \stackrel{\text{def}}{=} \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases} \tag{A.12}$$

For finding $\mathbf{w}_r$, the objective function J_r to be maximized is then given as:

$$J_r = \mathbf{w}_r^\top \mathbf{\Lambda} \mathbf{w}_r - \lambda_r \left(\mathbf{w}_r^\top \mathbf{w}_r - 1 \right) - \sum_{i=1}^{r-1} \mu_i \mathbf{w}_r^\top \mathbf{w}_i \tag{A.13}$$

where λ_r and μ_i are Lagrange multipliers. Note that the last term on the right side of Eq.(A.13) indicates that $\mathbf{y}_r$ is uncorrelated to $\mathbf{y}_i$ for all $i = 1, \ldots, r-1$. Setting the derivative of J_r equal to $\mathbf{0}$, we have:

$$\frac{\partial J_r}{\partial \mathbf{w}_r} = 2\mathbf{\Lambda} \mathbf{w}_r - 2\lambda_r \mathbf{w}_r - \sum_{i=1}^{r-1} \mu_i \mathbf{w}_i = \mathbf{0}. \tag{A.14}$$

From Eqs.(A.10), (A.11), and (A.14), we have $\mu_j = 0$ for $j = 1, \ldots, r-1$, and:

$$(\mathbf{\Lambda} - \lambda_r \mathbf{I})\, \mathbf{w}_r = \mathbf{0}. \tag{A.15}$$

Because the largest $r-1$ eigenvalues of $\mathbf{\Lambda}$ and the corresponding eigenvectors are already used for representing the first to $(r-1)$-th principal components, the variance of r-th principal

components is then given as the r-th largest eigenvalue, and the projection $\mathbf{w}_r$ as the corresponding eigenvector. Note that the matrix $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_r)^\top$ is orthonormal, and the data matrix $\mathbf{X} = \mathbf{W}^\top\mathbf{Y}$ can be factorized by the matrices $\mathbf{A} = \mathbf{W}^\top$ and $\mathbf{S} = \mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_r)^\top$ as in Eq.(A.1).

A.1.2 Singular Value Decomposition

The previous Section A.1.1 described that the principal components are highly related to the eigenvalue problem on the covariance matrix $\mathbf{\Lambda}$ of an appropriately standardized data matrix $\mathbf{X}$. Here we show that PCA is also highly related to the singular value decomposition (SVD), a powerful tool for data mining and linear regression (see below Section A.2).

As before, let us assume we have an $N \times M$ data matrix $\mathbf{X}$ containing M samples on the N-dimensional variables, and consider the following eigenvalue problem:

$$\frac{1}{M-1}\mathbf{X}\mathbf{X}^\top\mathbf{w}_i = \lambda_i\mathbf{w}_i \quad (i = 1, \ldots, N) \tag{A.16}$$

where λ_i and $\mathbf{w}_i$ are the i-th largest eigenvalue of the covariance matrix of $\mathbf{X}$ and its corresponding eigenvector normalized to have unit length (L_2-norm), respectively. We then have:

$$\mathbf{U}\mathbf{\Sigma} = \mathbf{X}^\top\mathbf{V}, \tag{A.17}$$

where

$$\mathbf{V} = \begin{pmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_N \end{pmatrix} = \mathbf{W}^\top \tag{A.18}$$

$$\sigma_i = \sqrt{(M-1)\lambda_i} \quad (i = 1, \ldots, N) \tag{A.19}$$

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_N \end{pmatrix} \tag{A.20}$$

$$\mathbf{u}_i = \frac{1}{\sigma_i}\mathbf{X}^\top \mathbf{w}_i \quad (i = 1, \ldots, N) \tag{A.21}$$

$$\mathbf{U} = \begin{pmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_N \end{pmatrix}. \tag{A.22}$$

Because the matrix $\mathbf{V}$ is orthonormal, we have the SVD of $\mathbf{X}$ from Eq.(A.17):

$$\mathbf{X} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top. \tag{A.23}$$

Note that the matrix $\mathbf{U}$ is also orthonormal:

$$\mathbf{u}_i^\top \mathbf{u}_j = \frac{1}{\sigma_i}\left(\mathbf{X}^\top \mathbf{w}_i\right)^\top \cdot \frac{1}{\sigma_j}\mathbf{X}^\top \mathbf{w}_j = \frac{1}{\sigma_i \sigma_j}\mathbf{w}_i^\top \mathbf{X}\mathbf{X}^\top \mathbf{w}_j = \delta_{ij}, \tag{A.24}$$

where the last equality holds from Eqs.(A.16) and (A.19). Suppose the rank of $\mathbf{X}$ is R, Eq.(A.23) can be written as:

$$\mathbf{X} = \sum_{r=1}^{R} \sigma_r \mathbf{w}_r \mathbf{u}_r^\top. \tag{A.23'}$$

Hence, the projection $\mathbf{w}_r$ for the r-th principal components can also be given as the left singular vector corresponding to the r-th largest singular value σ_r of $\mathbf{X}$.

A.1.3 Independent Component Analysis

While PCA seeks directions in "feature" space that best represent the data in a sum-of-squared-error sense, leading to *uncorrelated* directions, independent component analysis (ICA) seeks linear projections—not necessarily orthogonal to each other—that are most statistically *independent* from each other (Comon, 1994; Hyvärinen and Oja, 2000). Note that the latter is a much stronger condition than the former: uncorrelatedness involves only second-order statistics and implies independence only in the Gaussian case, whereas independence depends on all higher-order statistics and always implies uncorrelatedness.

One natural measure of the dependence between R random variables y_r (for $r = 1, \ldots, R$) is the mutual information (see also Section A.4):

$$I(y_1, \ldots, y_R) \stackrel{\text{def}}{=} \sum_{r=1}^{R} H(y_r) - H(\mathbf{y}), \tag{A.25}$$

where H is the differential entropy as defined in Eq.(A.80) on page 167. Note that the mutual information $I(y_1, \ldots, y_R)$ is equivalent to the Kullback-Leibler divergence between the joint density $p(\mathbf{y})$ and the product of its marginal densities $\prod_r p(y_r)$, and that Eq.(A.25) is always non-negative and equals zero if and only if the variables y_r are statistically independent.

Given an invertible linear transformation: $\mathbf{y} = \mathbf{W}\mathbf{x}$, Eq.(A.25) becomes:

$$I(y_1, \ldots, y_R) = \sum_{r=1}^{R} H(y_r) - H(\mathbf{x}) - \log|\det \mathbf{W}|, \tag{A.26}$$

where $|\det \mathbf{W}|$ is the absolute value of the determinant of $\mathbf{W}$. If we then constrain y_r to be *uncorrelated* and to have unit variance, we have:[1]

$$I(y_1, \ldots, y_R) = \sum_{r=1}^{R} H(y_r) + C_0 = -\sum_{r=1}^{R} Q(y_r) + C_0', \tag{A.27}$$

[1] In fact we will perform "whitening" on $\mathbf{x}$ and thus can constrain $\mathbf{W}$ to be orthogonal, so $|\det \mathbf{W}| = 1$.

where C_0 and C_0' are constants not dependent on $\mathbf{W}$, and $Q(\mathbf{y})$ is the "negentropy" of $\mathbf{y}$:

$$Q(\mathbf{y}) \stackrel{\text{def}}{=} H(\mathbf{y}_{\text{gauss}}) - H(\mathbf{y}), \tag{A.28}$$

where $\mathbf{y}_{\text{gauss}}$ is a set of Gaussian random variables that has the same covariance matrix as $\mathbf{y}$. Because a Gaussian variable has the largest entropy among all random variables of equal variance (see Eq.(A.83) on page 168), the negentropy is non-negative and can be considered as a natural measure of *non-gaussianity*. Hence, maximizing independence between y_r is equivalent to minimizing the mutual information $I(y_1, \ldots, y_R)$, which in turn is equivalent to maximizing the sum of the negentropies of y_r, i.e., the non-gaussianity.

One problem of using mutual information or negentropy as objective functions is that they are hard to estimate. Several approximation methods based on polynomials or higher-order statistics have then been proposed, and below I will describe one such algorithm for illustration—called "FastICA"—that exploits the fourth-order cumulant or kurtosis to maximize the non-gaussianity of y_r (Hyvärinen and Oja, 1997, 2000).

Whitening

Whitening is a preprocessing step where the observed data $\mathbf{x}$ is linearly transformed into a new vector $\widetilde{\mathbf{x}}$ of unit variance and uncorrelated elements:

$$\mathbb{E}\big[\widetilde{\mathbf{x}}\widetilde{\mathbf{x}}^\top\big] = \mathbf{I}. \tag{A.29}$$

This is useful for some ICA algorithms as it makes the target $\mathbf{W}$ a "rotation" matrix (see below). Using SVD as in Eq.(A.23), we can decompose the data matrix into: $\mathbf{X} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top$. Then such transformation that satisfies Eq.(A.29) is given by $\mathbf{M} = \mathbf{\Sigma}^{-1}\mathbf{V}^\top$, i.e.,

$$\widetilde{\mathbf{x}} = \mathbf{M}\mathbf{x} \approx \mathbf{M}\mathbf{A}\mathbf{s}. \tag{A.30}$$

The last approximation is from Eq.(A.1) with $\mathbf{s}$ being a corresponding column of $\mathbf{S}$, and we can see that $\mathbf{B} = \mathbf{MA}$ is orthogonal:

$$\mathbf{I} = \mathbb{E}\big[\widetilde{\mathbf{x}}\widetilde{\mathbf{x}}^\top\big] \approx \mathbf{B}\,\mathbb{E}\big[\mathbf{s}\mathbf{s}^\top\big]\mathbf{B}^\top \approx \mathbf{B}\mathbf{B}^\top. \tag{A.31}$$

The last approximation holds because we assume independent sources, $\mathbf{s}$, and because the sign and magnitude of the sources are arbitrary. Therefore, a goal in ICA can now be considered as to find appropriate orthogonal projection $\mathbf{W} = \mathbf{B}^\top$ on the whitened data $\widetilde{\mathbf{X}}$.

FastICA algorithm

The fourth-order cumulant—or kurtosis—of y is defined as:

$$\mathbb{K}\big[y\big] \stackrel{\text{def}}{=} \mathbb{E}\big[y^4\big] - 3\left(\mathbb{E}\big[y^2\big]\right)^2. \tag{A.32}$$

The kurtosis is zero for a Gaussian variable, positive for heavy-tailed super-Gaussian distributions, and negative for light-tailed sub-Gaussian distributions.

Using the Lagrange multiplier λ_1, the objective function for the first projection $y_1 = \mathbf{w}_1^\top\widetilde{\mathbf{x}}$ becomes:

$$J_1 = \mathbb{K}\big[y_1\big] + \lambda_1\left(\|\mathbf{w}_1\|^2 - 1\right) = \mathbb{E}\left[\left(\mathbf{w}_1^\top\widetilde{\mathbf{x}}\right)^4\right] - 3\|\mathbf{w}_1\|^4 + \lambda_1\left(\|\mathbf{w}_1\|^2 - 1\right). \tag{A.33}$$

Setting the derivative of J_1 equal to zero:

$$\frac{\partial J_1}{\partial \mathbf{w}_1} = \mathbb{E}\left[4\left(\mathbf{w}_1^\top\widetilde{\mathbf{x}}\right)^3\widetilde{\mathbf{x}}\right] - 12\|\mathbf{w}_1\|^2\mathbf{w}_1 + 2\lambda_1\mathbf{w}_1 = \mathbf{0}, \tag{A.34}$$

the update rule based on the fixed-point algorithm is then given as:

$$\mathbf{w} \leftarrow \frac{-2}{\lambda_1}\left(\mathbb{E}\left[\left(\mathbf{w}_1^\top\widetilde{\mathbf{x}}\right)^3\widetilde{\mathbf{x}}\right] - 3\|\mathbf{w}_1\|^2\mathbf{w}_1\right). \tag{A.35}$$

Considering the normalization $\|\mathbf{w}_1\| = 1$ from Eq.(A.3), we then have:

$$\mathbf{w}_1 \leftarrow \mathsf{normal}\left(\mathbb{E}\left[\left(\mathbf{w}_1^\top \widetilde{\mathbf{x}}\right)^3 \widetilde{\mathbf{x}}\right] - 3\mathbf{w}_1\right), \tag{A.36}$$

where $\mathsf{normal}(\cdot)$ is the operator for column-wise normalization.

For finding more than one independent projections $\mathbf{w}_r$ (for $r = 2, \ldots, R$), we can use the same update rule as Eq.(A.36) but must decorrelate $\mathbf{w}_r$ with all the other projections $\mathbf{w}_{r'}$ (for $r' = 1, \ldots, r-1$). Using $\mathbf{w}_r^\top \mathbf{w}_{r'} = \delta_{rr'}$, the Gram-Schmidt-like decorrelation gives the following additional step after every iteration by Eq.(A.36):

$$\mathbf{w}_r \leftarrow \mathbf{w}_r - \mathbf{B}_{r-1}\mathbf{B}_{r-1}^\top \mathbf{w}_r, \tag{A.37}$$

where $\mathbf{B}_{r-1} = (\mathbf{w}_1, \ldots, \mathbf{w}_{r-1})$. Finally, let $\mathbf{B} = \mathbf{B}_R$, and we have independent components $\mathbf{Y}$ from Eqs.(A.2) and (A.30):

$$\mathbf{Y} = \mathbf{B}^\top \widetilde{\mathbf{X}} = \mathbf{B}^\top \mathbf{M}\mathbf{X}. \tag{A.38}$$

Note that $\mathbf{Y}$ can be given by rotating the principal components of $\mathbf{X}$, i.e., $\mathbf{M}\mathbf{X} = \mathbf{U}^\top$, by the orthogonal matrix $\mathbf{B}^\top$.

It should also be mentioned that this FastICA algorithm is equivalent to replacing $-\log p(y_r)$ with y_r^4 in the definition of differential entropy ($H(y_r) \stackrel{\text{def}}{=} \mathbb{E}\big[-\log p(y_r)\big]$; Eq.(A.80) on page 167) used in the information theoretic objective in Eq.(A.27) on page 141, and seeking an orthogonal transformation $\mathbf{W}$ that gives an extremum on the fourth-order moment: $\mathbb{E}\big[y_r^4\big]$. Taking the first projection $\mathbf{w}_1$, for example, the objective function using the Lagrange multiplier λ_1 is:

$$J = \mathbb{E}\big[y_1^4\big] - \frac{\lambda_1}{2}\big(\mathbf{w}_1^\top \mathbf{w}_1 - 1\big). \tag{A.39}$$

Setting the derivative of J equal to zero, we have:

$$\frac{\partial J}{\partial \mathbf{w}_1} = \mathbb{E}\big[4\widetilde{\mathbf{x}} y_1^3\big] - \lambda_1 \mathbf{w}_1 = \mathbf{0}. \tag{A.40}$$

Using the Newton-Raphson method, the update rule to solve Eq.(A.40) is:

$$\mathbf{w}_1 \leftarrow \mathbf{w}_1 - \mathbf{f}'(\widetilde{\mathbf{x}})^{-1}\,\mathbf{f}\,(\widetilde{\mathbf{x}})\,, \quad \text{where} \tag{A.41}$$

$$\mathbf{f}\,(\widetilde{\mathbf{x}}) = \mathbb{E}\big[4\widetilde{\mathbf{x}}y_1^3\big] - \lambda_1\mathbf{w}_1, \tag{A.42}$$

$$\mathbf{f}'(\widetilde{\mathbf{x}}) = \mathbb{E}\big[12\widetilde{\mathbf{x}}\widetilde{\mathbf{x}}^\top y_1^2\big] - \lambda_1\mathbf{I} \;\approx\; \mathbb{E}\big[\widetilde{\mathbf{x}}\widetilde{\mathbf{x}}^\top\big]\,\mathbb{E}\big[12y_1^2\big] - \lambda_1\mathbf{I} \;\approx\; (12-\lambda_1)\mathbf{I}. \tag{A.43}$$

Note the whitening as in Eq.(A.29), and the constraint on $y_1 = \mathbf{w}_1^\top\widetilde{\mathbf{x}}$ to be uncorrelated and to have unit variance for obtaining the objective as in Eq.(A.27). Hence, substituting Eqs.(A.42) and (A.43) into Eq.(A.41) followed by (the multiplication with $\lambda_1/4-3$ and) the normalization on $\mathbf{w}_1$, we have the same update rule as Eq.(A.36).

A.1.4 Non-negative Matrix Factorization

Non-negative matrix factorization (NMF) achieves the approximation of a data matrix $\mathbf{X}$ as in Eq.(A.1) on page 136 under non-negativity constrains on each element of the factors $\mathbf{A}$ and $\mathbf{S}$ (Lee and Seung, 1999, 2001). With the Lagrange multiplier λ, the objective function to be minimized is given as:

$$J_{\text{NMF}} = \mathcal{D}(\mathbf{X}, \mathbf{AS}) + \lambda \cdot \text{regularizer} \quad \text{for} \quad \mathbf{A} \geq \mathbf{0}, \quad \mathbf{S} \geq \mathbf{0}, \tag{A.44}$$

where $\mathcal{D}(\cdot,\cdot)$ is a distance measure such as (matrix) norms or Kullback-Leibler divergence, and the regularizer can be, e.g., a sparseness constraint on $\mathbf{S}$: $\|\mathbf{S}\|_1 = \sum_{rm}|s_{rm}|$, or a smoothness constraint on $\mathbf{A}$: $\sum_{n,r}|a_{n,r} - a_{n-1,r}|$. For simplicity, however, here we ignore the regularizer (or $\lambda = 0$; see also Section A.2.1), and show below an algorithm that alternatively updates $\mathbf{A}$ and $\mathbf{S}$ where $\mathcal{D}(\cdot,\cdot)$ is the squared Frobenius norm (Lee and Seung, 2001). Taking the gradient of J_{NMF} with respect to $\mathbf{S}$:

$$\nabla_{\mathbf{S}} J_{\text{NMF}} = 2\mathbf{A}^\top(\mathbf{AS} - \mathbf{X})\,, \tag{A.45}$$

and setting a step-size matrix as:

$$\boldsymbol{\Delta}_{\mathbf{S}} = \frac{\mathbf{S}}{2\mathbf{A}^{\top}\mathbf{A}\mathbf{S}}, \tag{A.46}$$

a steepest descent gives the following update rule for $\mathbf{S}$:

$$\mathbf{S} \leftarrow \mathbf{S} - \boldsymbol{\Delta}_{\mathbf{S}} \bullet \nabla_{\mathbf{S}} J_{\text{NMF}} = \mathbf{S} \bullet \frac{\mathbf{A}^{\top}\mathbf{X}}{\mathbf{A}^{\top}\mathbf{A}\mathbf{S}}, \tag{A.47}$$

where the operators $\bullet$ and $\frac{\cdot}{\cdot}$ indicate elementwise multiplication and division, respectively. Similarly, the update rule for $\mathbf{A}$ can be written as:

$$\mathbf{A} \leftarrow \mathsf{normal}\left(\mathbf{A} \bullet \frac{\mathbf{X}\mathbf{S}^{\top}}{\mathbf{A}\mathbf{S}\mathbf{S}^{\top}}\right). \tag{A.48}$$

Note that the normalization of the columns of $\mathbf{A}$ to unit length is required for finding a unique solution, and that Eqs.(A.47) and (A.48) converge to a local minimum of Eq.(A.44).

A.1.5 Characteristics of Each Factorization Method

The three factorization methods described above (PCA, ICA, and NMF) all decompose a data matrix $\mathbf{X}$ into a new basis matrix $\mathbf{A}$ and a corresponding coefficient matrix $\mathbf{S}$, but have different characteristics because of the different constraints imposed on $\mathbf{A}$ and $\mathbf{S}$.

PCA constrains the columns of $\mathbf{A}$ to be orthonormal and the rows of $\mathbf{S}$ to be orthogonal. The resulting basis set $\mathbf{A}$ then has a good statistical interpretation as the subspace of largest variance, but it is often difficult to qualitatively interpret $\mathbf{A}$ because PCA allows negative values on $\mathbf{A}$ and $\mathbf{S}$.

ICA chooses the basis not as orthogonal but as independent as possible. This technique is thus more powerful than PCA to capture the statistical structures of the sources. Note however that, rather than directly optimize the *independence*, ICA algorithms typically maximize higher-order moments (e.g., kurtosis for FastICA in Section A.1.3), and thus the resulting components are not necessarily independent. Nevertheless, ICA is quite successful in many

cases—e.g., in the case of natural images and sounds (Simoncelli and Olshausen, 2001)—and has been applied to solve difficult problems such as the BSS problem (Choi et al., 2005).

In contrast to PCA and ICA, NMF does not allow negative entries in $\mathbf{A}$ and $\mathbf{S}$. Therefore, only additive combinations are allowed for data reconstruction, leading to a "parts"-based representation. (To my knowledge, however, no mathematical explanation is given for why this would be the case.) That is, NMF can even extract correlated features, and the basis often has intuitive interpretations.

Below I will show simulation results (done in MATLAB) where PCA, ICA, and NMF were applied to sample data sets such as *Kanji* Chinese characters to see the characteristics of each matrix decomposition method.

Example 1: *Kanji* Chinese characters

A data set of *kanji* Chinese characters was generated by rasterizing the "MS gothic" fonts (30 points), where some characters consist of combinations of "left" and "right" parts whereas others "top" and "bottom" parts. The data set contains images (30×30 pixels) of 1006 characters that students in Japan are supposed to learn by the age of 12 (National Curriculum Standards for Elementary Schools in Japan issued by the Ministry of Education, Culture, Sports, Science, and Technology; as of July, 2007). The list[2] was downloaded from http://www.mext.go.jp/b_menu/shuppan/sonota/990301b/990301d.htm (in Japanese).

When applied to the data set, NMF extracted those "meaningful" parts that are used frequently in the original data set, whereas PCA and ICA failed to find any "meaningful" basis elements but learned holistic representations (Figure A.1; data not shown for PCA).

[2] Also available at the website of AOZORA-BUNKO: http://www.aozora.gr.jp/kanji_table/kyouiku_list.zip.

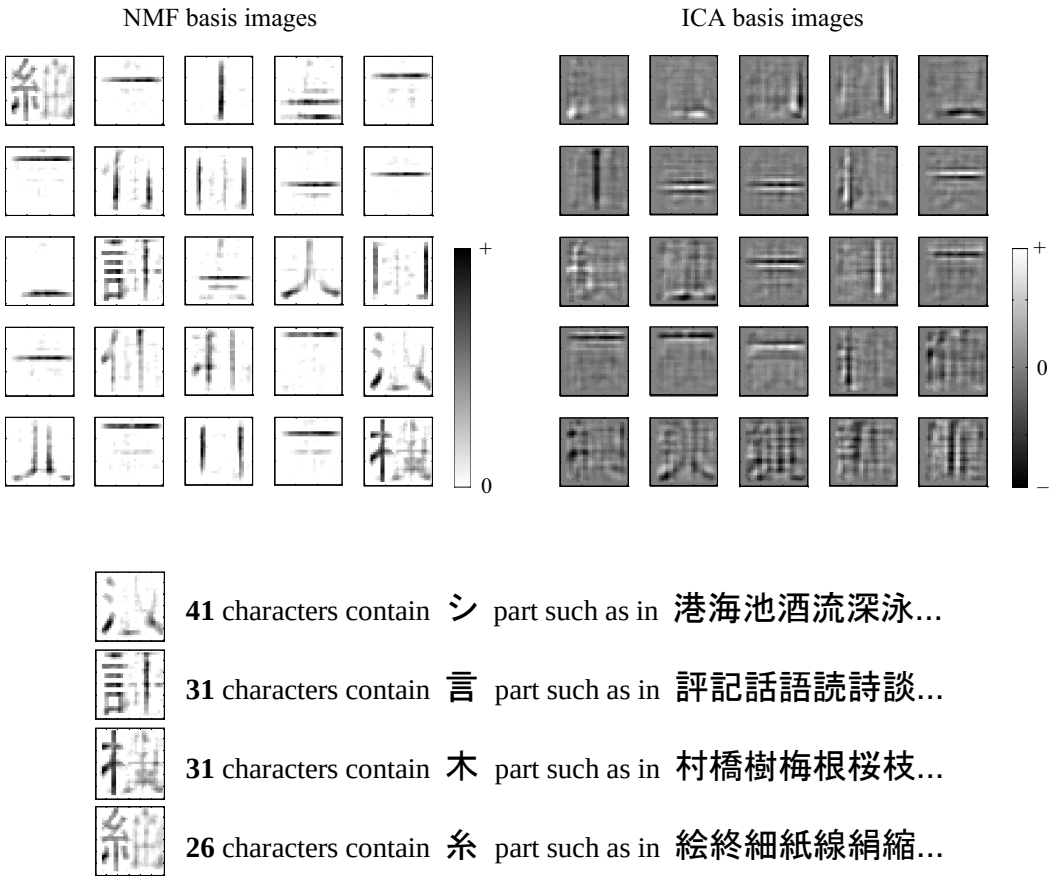

Figure A.1: **NMF found "meaningful" parts in Chinese characters while ICA failed**. NMF and ICA were applied to a data set of 1006 *kanji* Chinese characters (30×30 pixels). Some of the basis images obtained by NMF showed "meaningful" parts as shown in the bottom panel. In contrast, ICA learned a holistic representation.

Example 2: Images comprised of non-overlapping arbitrary basis

To confirm if NMF can always extract *the* basis images for the original data set, a set of 1000 images (25×25 pixel) were generated by an additive combination of 25 arbitrary non-overlapping basis images $\mathbf{A}_{\text{ini}}$ using the coefficients drawn from $|\mathcal{N}[0,1]|$, where $\mathcal{N}[\mu,\sigma^2]$ is the Gaussian—or Normal—distribution with mean μ and variance σ^2. The basis images $\mathbf{A}_{\text{ini}}$ were generated by the two-dimensional Normal distribution $\mathcal{N}[(\mu_x,\mu_y),\sigma^2]$ with mean $(\mu_x,\mu_y)=(5i-2,5j-2)$ for $i,j=1,2,3,4,5$ and standard deviation $\sigma=2.5$ pixels. Each basis image was then normalized to sum to unity, resulting in a not completely but essentially non-overlapping basis image set.

When the correct rank of factorization—i.e., the exact number of the underlying basis images ($R=25$ in this example)—was chosen, the basis images obtained by NMF had one-to-one correspondence to the original basis $\mathbf{A}_{\text{ini}}$ (Figure A.2). When the rank was chosen by far smaller or larger than the correct value, however, the obtained basis images showed quite different patterns from $\mathbf{A}_{\text{ini}}$. In contrast, when the rank was chosen a little smaller or larger than the correct rank size, some images showed redundancy but the others remained to have good one-to-one correspondence to $\mathbf{A}_{\text{ini}}$. These results show that it is important to choose an appropriate rank size R for the factorization, although it remains to be addressed what a good criterion would be. For data compression, R should satisfy: $NR+RM<NM$, but this inequality would not be very helpful for determining an appropriate rank size.

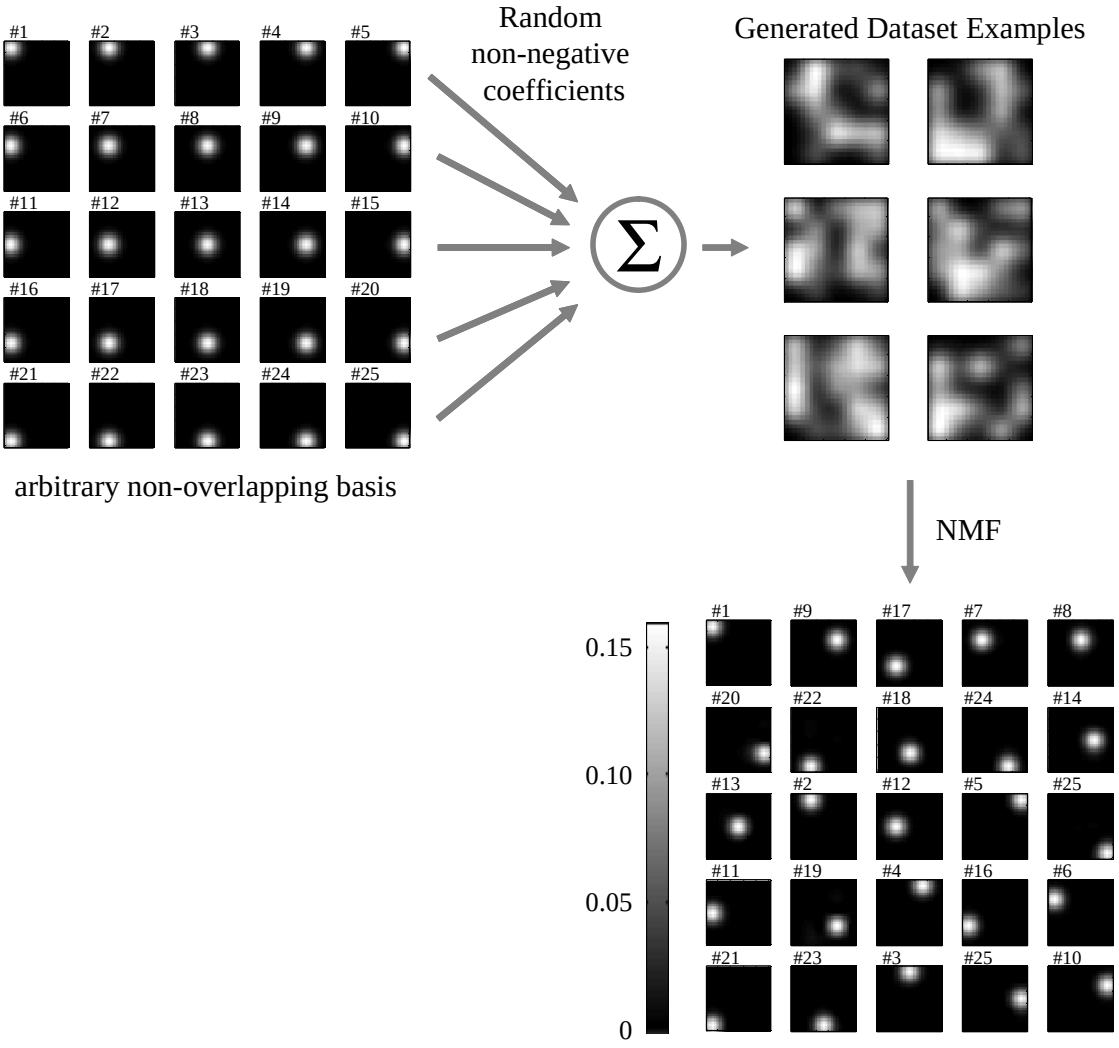

Figure A.2: **The original non-overlapping basis images and those obtained by NMF**. A set of 25 non-overlapping arbitrary basis images ($\mathbf{A}_{\text{ini}}$; *top-left*) was randomly combined to generate 1000 image data set (*top-right*), which in turn was decomposed back into basis images by NMF (*bottom-right*). Note that there is one-to-one correspondence between the two basis sets. The numbering was done by hand.

Example 3: Images comprised of overlapping arbitrary basis

The factorization methods was next tested on data sets $\mathbf{X}$ (1000 samples) generated by additive *dense* combinations of arbitrary overlapping basis images $\mathbf{A}_0$ (25×25 pixels) weighted with $|\mathcal{N}[0,1]|$, and on data sets $\mathbf{X}_S$ (1000 samples) generated by additive *sparse* combinations of randomly-chosen 10 out 25 basis images with the weights of $|\mathcal{N}[0,1]|$. To make the overlapping basis images $\mathbf{A}_0$, two random points were first chosen for each basis image within a two-dimensional plane (x, y) for both x and $y \in [0, 25]$ pixels. A line segment was generated by connecting these two points, and the value for each pixel on the plane was determined by: $\exp[-d^2/2]$, where d is the Euclidean distance between the line segment and the pixel of interest, followed by the normalization to sum to unity. In total, we generated 25 overlapping basis images $\mathbf{A}_0$ for simulations (Figure A.3, *top-left*).

Figure A.3 shows that NMF with the correct rank size ($R = 25$) worked well to extract underlying features $\mathbf{A}_0$ from the sparse data set $\mathbf{X}_S$ but not from the dense data set $\mathbf{X}$. In contrast, PCA and ICA failed to break even $\mathbf{X}_S$ back into the original basis $\mathbf{A}_0$ and corresponding coefficients, although some of the ICA basis images had one-to-one correspondence to $\mathbf{A}_0$. These simulation results then suggest that NMF would be preferable to ICA or PCA for extracting features of sound signals because natural sound ensembles are supposedly made up of a sparse combination of sound elements (see Section 2.2.4).

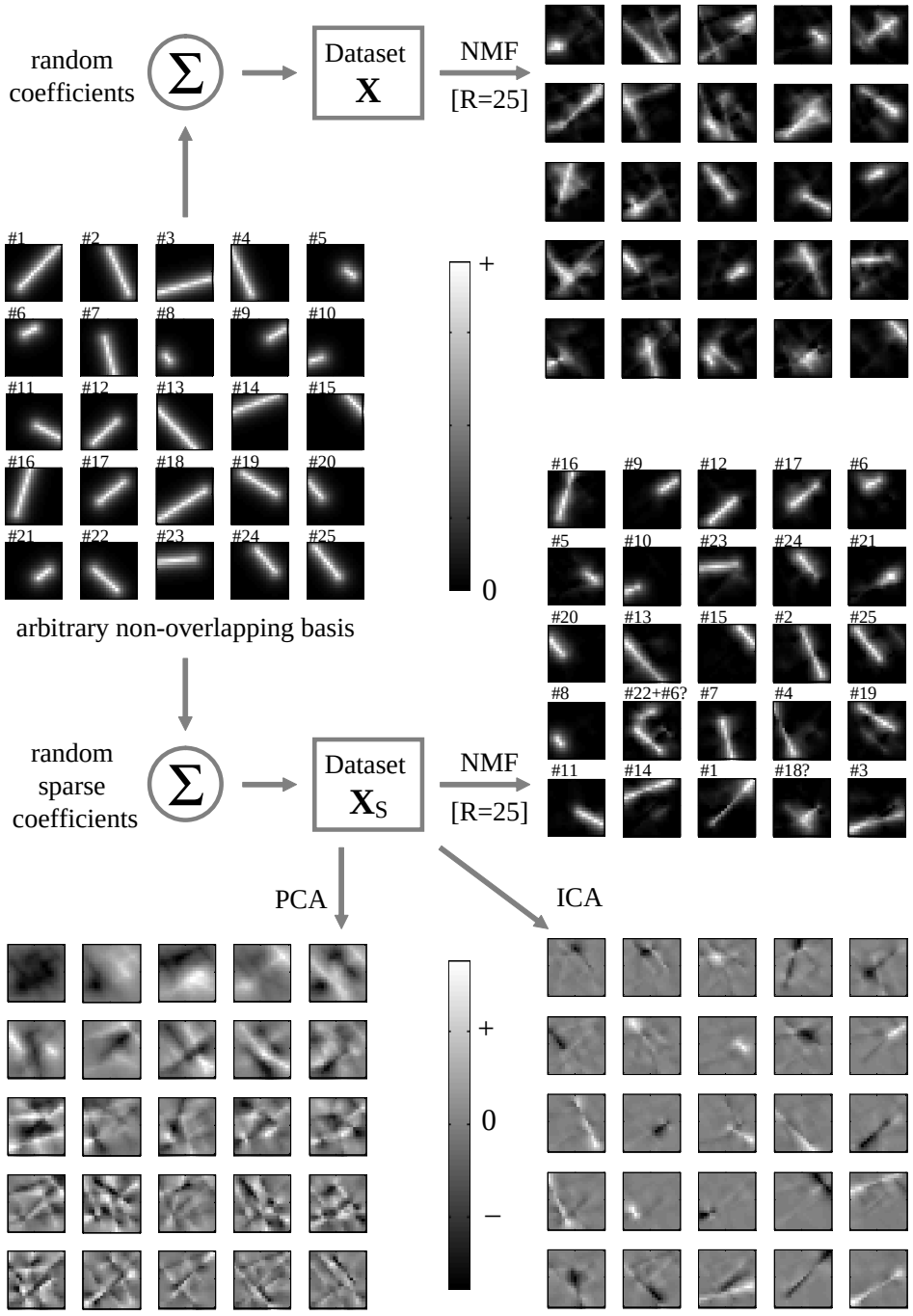

Figure A.3: **Original overlapping basis images $\mathbf{A}_0$ and those obtained by NMF, PCA, and ICA**. The basis images obtained by NMF from a sparse data set $\mathbf{X}_S$ showed good one-to-one correspondence to the original basis $\mathbf{A}_0$ (except for #18), while the other methods failed to show good fidelity.

A.2 Linear Regression

To understand how the brain processes sensory information, it is important to study the relationship between input stimuli and the output neural responses (for review, see e.g., Simoncelli et al., 2004; Wu et al., 2006). Here I briefly overview linear regression, one of the basic and the simplest methods for modeling input-output functions. Section A.2.1 reviews regularization methods to find the best (static) linear models (for details, see e.g., Hastie et al., 2001; Duda et al., 2001; Chen and Haykin, 2002), and Section A.2.2 describes the regression techniques from the probabilistic viewpoints. Section A.2.3 then shows some ways for incorporating temporal dynamics to better characterize the system (for further extensions, see e.g., Nelles, 2001).

A general goal in a regression model is to predict an output y from a vector of inputs $\mathbf{x} \in \mathbb{R}^N$. The linear regression model assumes that the regression function f_{L} is linear[3] and has the form:

$$\widehat{y} = f_{\mathrm{L}}(\mathbf{x}) = \alpha_0 + \sum_{n=1}^{N} \alpha_n x_n = \alpha_0 + \mathbf{x}^\top \boldsymbol{\alpha} \tag{A.49}$$

where $\widehat{y}$ is the estimated output, and $\boldsymbol{\alpha}$ (and α_0) are unknown parameters or coefficients. Typically we have a set of training data: $(y_m, \mathbf{x}_m)$ for $m = 1, \ldots, M\,(\geq N)$ to estimate the coefficients $\boldsymbol{\alpha}$ (and α_0). The most common estimation method is to minimize the residual sum of squared errors between the estimated output $\widehat{\mathbf{y}}$ and the original output $\mathbf{y}$:

$$E(\boldsymbol{\alpha}) = \|\mathbf{y} - \widehat{\mathbf{y}}\|^2 = \left(\mathbf{y} - \mathbf{X}^\top \boldsymbol{\alpha}\right)^\top \left(\mathbf{y} - \mathbf{X}^\top \boldsymbol{\alpha}\right), \tag{A.50}$$

where the m-th column of the matrix $\mathbf{X}$ consists of an m-th input vector $\mathbf{x}_m$. For simplicity, here we assume that the outputs have zero mean: $\sum_m y_m = 0$, that is, $\alpha_0 = 0$ in Eq.(A.49).

[3]For a member of any vector space x (and y), a function $f(x)$ is linear if it satisfies $f(x+y) = f(x) + f(y)$ and $f(\alpha x) = \alpha f(x)$ for all scalar α.

The least square solution is then given by:

$$\widehat{\boldsymbol{\alpha}}_{\text{ls}} = \left(\mathbf{X}\mathbf{X}^\top\right)^{-1}\mathbf{X}\mathbf{y}. \tag{A.51}$$

Note that $\mathbf{X}^\dagger = \left(\mathbf{X}\mathbf{X}^\top\right)^{-1}\mathbf{X}$ is called the pseudoinverse of $\mathbf{X}^\top$, and that $\mathbf{X}\mathbf{X}^\top$ and $\mathbf{X}\mathbf{y}$ are sometimes referred as *auto-correlation* and *cross-correlation*, respectively, and Eq.(A.51) as *reverse correlation* in the neurophysiological jargon (de Boer and Kuyper, 1968; Eggermont et al., 1983; Ringach and Shapley, 2004).

A.2.1 Regularization Methods

In practice, the auto-correlation $\mathbf{X}\mathbf{X}^\top$ in Eq.(A.51) could have some eigenvalues close to zero, leading to an overfitting and a very noisy estimate of the coefficients $\boldsymbol{\alpha}$. To address this issue,[4] a regularizer is often introduced to place constraints on the coefficients so that we do not suffer much from high variability in the estimation. The cost function to be minimized is then, from Eq.(A.50),

$$E_{\text{reg}}(\boldsymbol{\alpha}, \lambda) = E(\boldsymbol{\alpha}) + \lambda \cdot \text{regularizer}, \tag{A.52}$$

where the parameter $\lambda \geq 0$ determines the strength of the constraint.

Ridge regression is one of the shrinkage methods to penalize strong deviations of the parameters from zero; i.e., the "regularizer" in Eq.(A.52) is given as: $\|\boldsymbol{\alpha}\|_2^2 = \boldsymbol{\alpha}^\top\boldsymbol{\alpha}$. The solution for ridge regression can then be expressed as:

$$\widehat{\boldsymbol{\alpha}}_{\text{ridge}} = \left(\mathbf{X}\mathbf{X}^\top + \lambda\,\mathbf{I}\right)^{-1}\mathbf{X}\mathbf{y}. \tag{A.53}$$

[4] Another common technique is to weight the contribution of the square residuals on the cost function for each data; i.e., to minimize $E_{\text{wls}}(\boldsymbol{\alpha}, \boldsymbol{\Omega}) = (\mathbf{y} - \mathbf{X}^\top\boldsymbol{\alpha})^\top\boldsymbol{\Omega}(\mathbf{y} - \mathbf{X}^\top\boldsymbol{\alpha})$, where $\boldsymbol{\Omega}$ is a diagonal matrix whose (i,i)-element consists of the weight ω_i on the i-th data. The solution of weighted least squares is given as: $\widehat{\boldsymbol{\alpha}}_{\text{wls}} = (\mathbf{X}\boldsymbol{\Omega}\mathbf{X}^\top)^{-1}\mathbf{X}\boldsymbol{\Omega}\mathbf{y}$, and $\widehat{\boldsymbol{\alpha}}_{\text{wls}} = \widehat{\boldsymbol{\alpha}}_{\text{ls}}$ when $\boldsymbol{\Omega} = \mathbf{I}$. Note that $\boldsymbol{\Omega}$ has many free parameters to be estimated, and thus the regularizer approach would be preferable that often has less free parameters; e.g., ridge regression in Eq.(A.53) has only a single parameter λ.

Note that the solution adds a positive constant λ to the diagonal of $\mathbf{X}\mathbf{X}^\top$ before the inverse, which makes the matrix well conditioned (nonsingular) even if $\mathbf{X}\mathbf{X}^\top$ is poorly conditioned (or is not practically a full-rank matrix). Also note that the ridge parameter λ can be determined by common model selection techniques (Hastie et al., 2001; MacKay, 2003), such as Akaike or Bayesian information criteria (Akaike, 1974; Schwarz, 1978), minimal description length (Rissanen, 1983, 1986), or (generalized) cross-validation (Stone, 1974; Efron, 1983).

The least square solution $\widehat{\boldsymbol{\alpha}}_{\text{ls}}$ in Eq.(A.51) and the ridge regression solution $\widehat{\boldsymbol{\alpha}}_{\text{ridge}}$ in Eq.(A.53) are highly related to the SVD described in Section A.1.2. Following the notation in Eq.(A.23) on page 140, the pseudoinverse $\mathbf{X}^\dagger$ can be written as:

$$\mathbf{X}^\dagger = \left(\mathbf{X}\mathbf{X}^\top\right)^{-1}\mathbf{X} = \left(\mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^\top\right)^{-1}\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^\top = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^\top. \tag{A.54}$$

Therefore, from Eqs.(A.51) and (A.53),

$$\widehat{\boldsymbol{\alpha}}_{\text{ls}} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^\top\mathbf{y}, \tag{A.51'}$$

$$\widehat{\boldsymbol{\alpha}}_{\text{ridge}} = \mathbf{V}\left(\boldsymbol{\Sigma}^2 + \lambda\,\mathbf{I}\right)^{-1}\boldsymbol{\Sigma}\mathbf{U}^\top\mathbf{y}. \tag{A.53'}$$

Note that the (r,r)-element of the diagonal matrix $\left(\boldsymbol{\Sigma}^2 + \lambda\,\mathbf{I}\right)^{-1}\boldsymbol{\Sigma}$ is $\sigma_r/(\sigma_r^2+\lambda)$, where σ_r is the r-th largest singular value. The estimated outputs $\widehat{\mathbf{y}} = \mathbf{X}^\top\widehat{\boldsymbol{\alpha}}$ can then be written as:

$$\widehat{\mathbf{y}}_{\text{ls}} = \mathbf{X}^\top\left(\mathbf{X}\mathbf{X}^\top\right)^{-1}\mathbf{X}\mathbf{y} = \mathbf{U}\mathbf{U}^\top\mathbf{y}, \tag{A.55}$$

$$\widehat{\mathbf{y}}_{\text{ridge}} = \mathbf{X}^\top\left(\mathbf{X}\mathbf{X}^\top + \lambda\,\mathbf{I}\right)^{-1}\mathbf{X}\mathbf{y} = \mathbf{U}\boldsymbol{\Sigma}\left(\boldsymbol{\Sigma}^2 + \lambda\,\mathbf{I}\right)^{-1}\boldsymbol{\Sigma}\mathbf{U}^\top\mathbf{y} = \sum_r \mathbf{u}_r\frac{\sigma_r^2}{\sigma_r^2+\lambda}\mathbf{u}_r^\top\mathbf{y}. \tag{A.56}$$

Note that $\mathbf{U}^\top\mathbf{y}$ in least squares (Eq.(A.55)) are the coordinates of $\mathbf{y}$ with respect to the orthogonal basis $\mathbf{U}$, and the coordinates are shrunk by the factor of $\sigma_r^2/(\sigma_r^2+\lambda)$ in the ridge regression (Eq.(A.56)). In least squares for example, the estimation noise is then given as:

$$\boldsymbol{\eta}_{\text{ls}} = \mathbf{y} - \widehat{\mathbf{y}}_{\text{ls}} = \left(\mathbf{I} - \mathbf{U}\mathbf{U}^\top\right)\mathbf{y}. \tag{A.57}$$

A.2.2 Probabilistic Interpretation

The regression framework described above is based on a frequentist viewpoint, but it can be readily interpreted from a probabilistic or Bayesian viewpoint as well. Suppose that a sample y is derived from a joint density $p(y, \theta)$—or *likelihood*—with some parameters θ, the principle of maximum likelihood (ML) assumes that the most reasonable values for θ are the ones for which the probability of obtaining the observed samples is the largest.

Least squares regression is then the same as ML under the assumption of additive i.i.d. Gaussian noise. Using the conditional likelihood: $p(y|\mathbf{x}, \theta) = \mathcal{N}\big[\mathbf{x}^\top \boldsymbol{\alpha}, \sigma^2\big]$, where $\theta \equiv \{\boldsymbol{\alpha}, \sigma\}$ for the linear regression as in Eq.(A.49), the log-likelihood of the observed data set $(y_m, \mathbf{x}_m)$ for $m = 1, \ldots, M$ can be written as, from Eq.(A.50),

$$\sum_{m=1}^{M} \log p(y_m|\mathbf{x}_m, \theta) = -\frac{E(\boldsymbol{\alpha})}{2\sigma^2} - M \log \sqrt{2\pi\sigma^2}. \tag{A.58}$$

Maximizing Eq.(A.58) is thus equivalent to minimizing $E(\boldsymbol{\alpha})$; i.e., $\widehat{\boldsymbol{\alpha}}_{\text{ML}} = \widehat{\boldsymbol{\alpha}}_{\text{ls}}$ as in Eq.(A.51), and the unbiased estimate of σ^2 is given as, from Eq.(A.57): $(M - N - 1)\widehat{\sigma}^2 = \|\boldsymbol{\eta}_{\text{ls}}\|_2^2 \sim \sigma^2 \chi^2_{M-N-1}$, where χ^2_{M-N-1} is a chi-square distribution with $M - N - 1$ degrees of freedom. Different cost functions are obtained by the ML approach for other noise distributions.

Regularization methods in Section A.2.1 can be considered as maximum *a posteriori* (MAP) approach where the parameters θ in ML are also treated as random variables. Let $p(\theta)$ be a *prior* distribution of θ, then a *posterior* distribution $p(\theta|y)$ can be given by the Bayes theorem:

$$p(y, \theta) = p(\theta|y)p(y) = p(y|\theta)p(\theta), \quad \text{hence}, \quad \text{Posterior} \;\propto\; \text{Likelihood} \times \text{Prior}. \tag{A.59}$$

The prior can be written in the exponential form: $p(\theta) \propto \exp\big[-\lambda \cdot \text{regularizer}\big]$, without loss of generality. Taking the negative logarithm of Eq.(A.59), we then have the equivalent objective

as Eq.(A.52):

$$-\log p(\theta|y) \propto -\log p(y|\theta) - \log p(\theta) = E_{\text{reg}}(\boldsymbol{\alpha}, \lambda). \tag{A.60}$$

Therefore, MAP with a Gaussian prior is equivalent to ridge regression with an L_2 penalty on $\boldsymbol{\alpha}$, and in the case of a Laplacian prior, MAP is equivalent to *lasso* (least absolute shrinkage and selection operator) regression where an L_1 penalty is imposed on $\boldsymbol{\alpha}$, encouraging sparsity (see also Section 2.2.3). Note that ML (without regularizer) is a special case of MAP with a "flat" prior.

A.2.3 Dynamic Models

Because the encoding properties of neurons typically vary over time, static models such as Eq.(A.49) are often too simple to account for the neural behaviors in response to sensory signals, even though we have a reasonable interpretation; e.g., $\boldsymbol{\alpha}$ corresponds to the neuron's "receptive field" (Barlow, 1953; Kuffler, 1953; Ringach, 2004). Here I explain two ways to extend the static models by incorporating temporal dynamics.

Adaptive Filters

One common technique to incorporate temporal dynamics for extending the static models is to make the filter $\boldsymbol{\alpha}$ adaptive (Stanley, 2002):

$$\widehat{\boldsymbol{\alpha}}_m = \arg\min_{\boldsymbol{\alpha}} \sum_{t=0}^{T} \xi^t \big(y_{m-t} - \widehat{y}_{m-t}\big)^2, \tag{A.61}$$

where $\widehat{y}_m$ follows Eq.(A.49), and $\xi \in [0, 1]$ is a "forgetting factor" that down-weights past information in an exponential manner. Note that the estimation is based only on the past information up to time T, instead of using the whole data as in Eq.(A.51) or Eq.(A.53).

If T is large enough, the auto-correlation $\boldsymbol{\Phi}_m$ and the cross-correlation $\boldsymbol{\psi}_m$ at time m can be approximated as:

$$\boldsymbol{\Phi}_m = \sum_{t=0}^{T} \xi^t \mathbf{x}_{m-t} \mathbf{x}_{m-t}^\top \;\approx\; \xi \boldsymbol{\Phi}_{m-1} + \mathbf{x}_m \mathbf{x}_m^\top, \tag{A.62}$$

$$\boldsymbol{\psi}_i = \sum_{t=0}^{T} \xi^t \mathbf{x}_{m-t} y_{m-t} \;\approx\; \xi \boldsymbol{\psi}_{m-1} + \mathbf{x}_m y_m. \tag{A.63}$$

The adaptive filter $\widehat{\boldsymbol{\alpha}}_m$ in Eq.(A.61) can then be estimated as:

$$\widehat{\boldsymbol{\alpha}}_m = \big(\boldsymbol{\Phi}_m + \lambda \mathbf{I}\big)^{-1} \boldsymbol{\psi}_m \;\approx\; \widehat{\boldsymbol{\alpha}}_{m-1} + \big(\boldsymbol{\Phi}_m + \lambda \mathbf{I}\big)^{-1} \mathbf{x}_m \big(y_m - \mathbf{x}_m^\top \widehat{\boldsymbol{\alpha}}_{m-1}\big), \tag{A.64}$$

where the first equality is by the ridge regression technique (see above Section A.2.1). Note that the development of the adaptive filter consists of the estimate at time $m-1$ plus the update (error correction) based on the new information at time m.

Integrate-and-fire-like Model

Another way to extend the static model is to consider both input- and output-history dependence (up to time τ_x and τ_y, respectively), i.e., to incorporate a recursive term into Eq.(A.49):

$$\widehat{y}_m = \alpha_0 + \sum_{t=1}^{\tau_x} \mathbf{x}_{m-t}^\top \boldsymbol{\alpha}_t + \sum_{t=1}^{\tau_y} y_{m-t} \beta_t. \tag{A.65}$$

Eq.(A.65) implements an infinite impulse response (IIR) filter whereas the earlier formulations are all finite impulse response (FIR) filters, and that linear regression technique can be exploited to estimate the parameters $\boldsymbol{\alpha}_t$ and β_t after some rearrangement (Nelles, 2001). This recursive least squares method is essentially equivalent to the Kalman filter, although the latter is usually applied for the estimation of states rather than just parameters (Kalman, 1960a,b; Roweis and Ghahramani, 1999).

We could also rewrite Eq.(A.65) as follows:

$$\frac{\Delta y_m}{\Delta m} = \alpha_0 + \sum_{t=1}^{\tau_x} \mathbf{x}_{m-t}^{\top} \boldsymbol{\alpha}_t - \sum_{t=1}^{\tau_y} y_{m-t} \bar{\beta}_t, \tag{A.66}$$

which is a variant of an integrate-and-fire (IF) model (Lapicque, 1907; Stevens and Zador, 1998; Gerstner and Kistler, 2002; Paninski et al., 2005). The parameters could then be interpreted more readily from the viewpoint of neurons—as before, $\boldsymbol{\alpha}_t$ corresponds to the neuron's "receptive field," and $\bar{\beta}_t$ response-dependent factors such as refractoriness or adaptation effects—and more elaborate models have been proposed and applied in this context (see e.g., Fishbach et al., 2001; Keat et al., 2001; Gerstner and Kistler, 2002; Fishbach et al., 2003; Paninski et al., 2004; Pillow et al., 2005).

A.3 Nonlinear Regression

Linear models have been widely used in sensory systems neuroscience due to their simplicity and interpretability (Paninski, 2003a; Simoncelli et al., 2004; Wu et al., 2006). However, neural responses to sensory stimuli cannot be fully explained by such "simple" encoding models in general, even though neurons sometimes show high trial-to-trial reliability (for the primary auditory cortex, see e.g., Machens et al., 2004; Deweese and Zador, 2004; see also Chapter 3). We thus need more "complex" models, and below I will briefly overview some nonlinear regression techniques, especially from a viewpoint of artificial neural networks. Section A.3.1 shows that incorporating (static) nonlinearities into the linear models (see above Section A.2) leads to an equivalent model structure as a "perceptron." Section A.3.2 then describes a general framework—multi-layer neural networks—that has more "units/layers" and higher expressive power. Note that the central idea of the model framework is: (1) to extract features as linear combinations of inputs, and (2) to model outputs as a nonlinear function of these features. Finally, Section A.3.3 shows one technique—support vector regression—developed recently for obtaining better generalization performance.

A.3.1 Perceptron

(Static) nonlinearities can be given as a nonlinear transformation f that acts on the output of the linear model (f_{L} as in Eq.(A.49) on page 153) to form a new better estimate (see e.g., Paninski, 2003a; Simoncelli et al., 2004; Machens et al., 2004):

$$\widehat{y} = f\big(f_{\mathrm{L}}(\mathbf{x})\big) = f\big(\alpha_0 + \mathbf{x}^{\top}\boldsymbol{\alpha}\big). \tag{A.67}$$

For finding a nonlinear function f, we typically plot the actual output y against the linear estimates $f_{\mathrm{L}}(\mathbf{x})$ and generate a calibration curve, e.g., by robust locally weighted scatter-plot smoothing (LOWESS; Cleveland, 1979; Cleveland and Devlin, 1988). Note that this approach does not formally yield the optimal estimate of f when $\widehat{\boldsymbol{\alpha}}$ is a biased estimator due to data limitations and/or sampling biases, and simultaneous updates of f and f_{L}—e.g., to maximize mutual information between y and $f_{\mathrm{L}}(\mathbf{x})$ by gradient descent—are required in general for optimal estimation (Paninski, 2003a; Sharpee et al., 2004, 2006).

The model structure in Eq.(A.67) is known as a "neuron"—or a "perceptron"—in artificial neural network studies (McCulloch and Pitts, 1943; Rosenblatt, 1958), and has been widely applied in pattern classifications (Hertz et al., 1991; Hastie et al., 2001; Duda et al., 2001) as well as in neuroscience; e.g., cerebellar functions can be well explained as perceptrons (Marr, 1969; Albus, 1971; Ito et al., 1982). However, (single-layer) perceptrons work only for "linearly separable" problems (Minsky and Papert, 1969)—e.g., the logical function of exclusive-OR (XOR) cannot be learned—and "hidden layers" are required to overcome the limitations (see below Section A.3.2).

A.3.2 Artificial Neural Network

Multi-layer networks can provide optimal solutions to an arbitrary fitting problem (Hecht-Nielsen, 1989; Hertz et al., 1991), where each "unit" in each "layer" extracts features as linear combinations of inputs (from units in the previous layer) and passes outputs by a nonlinear

function of these features to units in the following layer. In the case of three-layer feedforward networks that consist of input, hidden, and output layers interconnected with modifiable weights, the regression function can be written as a cascade of mappings:

$$\widehat{y} = f\big(\alpha_0 + \mathbf{z}^\top \boldsymbol{\alpha}\big), \quad \text{where} \quad z_r = f_r\big(\alpha_{r0} + \mathbf{x}^\top \boldsymbol{\alpha}_r\big) \quad \text{for} \quad r = 1, \ldots, R. \tag{A.68}$$

Note that this formulation is in essence no different from the basic framework of Eq.(A.67); i.e., Eq.(A.68) performs linear regression with $\boldsymbol{\alpha}$—followed by nonlinear transformation f—in a "feature" space of $\mathbf{z} \in \mathbb{R}^R$ where the inputs $\mathbf{x} \in \mathbb{R}^N$ have been mapped nonlinearly by $\boldsymbol{\alpha}_r$ and f_r.

One of the most popular methods for training parameters in such artificial neural networks is known as "backpropagation" or the generalized delta rule (Widrow and Hoff, 1960; Rumelhart et al., 1986), based on gradient descent in the sum squared errors (see also least squares; Eq.(A.49)). Let us suppose we have a training data set: $(y_m, \mathbf{x}_m)$ for $m = 1, \ldots, M$, then the learning algorithm can be written as:

$$E(\alpha) = \sum_{m=1}^{M} \big(y_m - \widehat{y}_m\big)^2 \quad \text{and} \quad \alpha \leftarrow \alpha - \lambda \, \frac{\partial E}{\partial \alpha}, \tag{A.69}$$

where $\lambda (> 0)$ is the learning rate—indicating the relative size of the change in parameters or "weights"—and the gradient $\partial E / \partial \alpha$ can be computed by using the chain rule for differentiation:

$$\frac{\partial E}{\partial \alpha_r} = -\sum_{m=1}^{M} \big(y_m - \widehat{y}_m\big) \, f'\big(\alpha_0 + \mathbf{z}_m^\top \boldsymbol{\alpha}\big) \, z_{rm}, \tag{A.70}$$

$$\frac{\partial E}{\partial \alpha_{rn}} = -\sum_{m=1}^{M} \big(y_m - \widehat{y}_m\big) \, f'\big(\alpha_0 + \mathbf{z}_m^\top \boldsymbol{\alpha}\big) \, \alpha_r \, f_r'\big(\alpha_{r0} + \mathbf{x}_m^\top \boldsymbol{\alpha}_r\big) \, x_{nm}. \tag{A.71}$$

Let us write Eqs.(A.70) and (A.71) as:

$$\frac{\partial E}{\partial \alpha_r} = \sum_{m=1}^{M} \Delta_m^{\text{output}} \, z_{rm}, \tag{A.72}$$

$$\frac{\partial E}{\partial \alpha_{rn}} = \sum_{m=1}^{M} \Delta_{rm}^{\text{hidden}} \, x_{nm}, \tag{A.73}$$

where Δ_m^{output} and $\Delta_{rm}^{\text{hidden}}$ are the "errors"—or the "sensitivity"—from the current model at the output and hidden layer units, respectively. Then we have:

$$\Delta_{rm}^{\text{hidden}} = f_r'\big(\alpha_{r0} + \mathbf{x}_m^\top \boldsymbol{\alpha}_r\big) \, \alpha_r \, \Delta_m^{\text{output}}, \tag{A.74}$$

showing that the sensitivity is propagated "back" from the output unit to the hidden units.

Several comments for the application of neural networks are in order (for details, see e.g., Hertz et al., 1991; Hastie et al., 2001; Duda et al., 2001). First, the learning algorithm described above was the most straightforward one, and more sophisticated algorithms have been developed, e.g., by using conjugate gradient descent based on a second-order analysis of the error function. Second, common choices of (differentiable) functions f and f_r are Gaussian (radial basis), logistic/sigmoidal, polynomial, and Heaviside/rectification functions. Third, neural networks often have too many parameters and will easily overfit the data. Several regularization methods have been developed to overcome this drawback— e.g., early stoppings, cross-validation, complexity adjustment/pruning, and weight decay (which is essentially equivalent to ridge constraints as described in Section A.2.1), and so on—but it is also important to choose appropriate number of hidden units/layers and initial conditions to find optimal solutions. Note also that the error function often possesses many local minima. Finally, it is often useful to preprocess data—e.g., standardization or dimension reduction by PCA—before training a network.

In sensory systems neuroscience, artificial neural networks were applied to characterize neural response patterns (Lehky et al., 1992; Lau et al., 2002; Prenger et al., 2004), but

criticized for the lack of biological interpretability and plausibility (Grossberg, 1987; Stork, 1989) and thus less popular despite their high expressive power. However, it should be mentioned that most neural encoding/decoding models have the same structure as Eq.(A.68); i.e., cascades of linear and nonlinear transformations (Hunter and Korenberg, 1986; Korenberg and Hunter, 1986, 1996). Because the dimension of input stimuli is typically huge, we often search for the subspace—or feature space—relevant for neural behavior on the basis of, e.g., orientation, contrast, spatial frequency or phase for visual stimuli (e.g., Enrorh-Cugel and Robson, 1966; Movshon et al., 1978; Jones and Palmer, 1987; Tolhurst and Dean, 1990; Reid and Shapley, 1992; DeAngelis et al., 1993a,b; Ringach et al., 1997; Brenner et al., 2000; Touryan et al., 2002; David et al., 2004; Rust and Movshon, 2005), and frequency, spectrogram, or ripple domains for auditory stimuli (e.g., Klein et al., 2000; Theunissen et al., 2000, 2001; Depireux et al., 2001; Escabí and Schreiner, 2002; Machens et al., 2004). Such subspaces can then be interpreted as "tuning curves" and/or "receptive fields" of neurons of interest.

Models in neuroscience typically have a relatively small number of "hidden units," and many feature extraction—or dimension reduction—techniques have been proposed and applied in sensory systems research, such as spike triggered average/covariance (or white-noise analysis; Bussgang, 1952; de Boer and Kuyper, 1968; Marmarelis and Naka, 1972; Marmarelis and Marmerelis, 1978; Aertsen and Johannesma, 1981; Eggermont et al., 1983; Eggermont, 1993; Ringach and Shapley, 2004; Schwartz et al., 2002; Simoncelli et al., 2004), and maximally informative dimensions (Paninski, 2003a; Sharpee et al., 2004, 2006). The fundamental idea here is to find a subspace that can best differentiate the stimulus distribution of interest—such as those evoked spikes $p(x|\text{spike})$—from the whole stimulus distribution $p(x)$, and the nonlinearities can then be recovered by a direct comparison between the two distributions using Bayes rule ($p(\text{spike}|x) \propto p(x|\text{spike})/p(x)$; de Ruyter van Steveninck and Bialek, 1988; Pillow and Simoncelli, 2006), or by assuming some specific/judicious form of gain controls (e.g., divisive normalization; Schwartz and Simoncelli, 2001). However, it is generally very difficult to find out *a priori* what form of nonlinear transformations neurons would perform (but

see e.g., Adelson and Bergen, 1985; Victor, 1992, 2005; Chichilnisky, 2001; Fishbach et al., 2001, 2003; Keat et al., 2001; Nykamp and Ringach, 2002; Herz et al., 2006)—even if target phenomena could be explained faithfully, we should not overinterpret any model unless it is verified to be (most likely) the case by enough experimental evidences. We should remember that many classes of models could exist that all work well but have nothing to do with biology, and that we could be deceived by such "wrong"—or nonbiological—models.

A.3.3 Support Vector Regression

Support vector machine (SVM)—a technique based on the statistical learning theory and the Vapnik-Chervonenkis (VC) dimension (Vapnik and Chervonenkis, 1971)—is a powerful tool for data classification (Boser et al., 1992; Vapnik, 1995), and can be applied for regression, maintaining the main advantages of the support vector methods (for tutorial, see Smola and Schölkopf, 2004). The basic idea is to map the inputs $\mathbf{x}$ into a high-dimensional space by nonlinear transformations f_r, and perform linear regression in this feature space. Formally, we have:

$$\widehat{y} = \alpha_0 + \sum_{r=1}^{R} \alpha_r \, f_r(\mathbf{x}). \tag{A.75}$$

The goal in "epsilon-support vector regression (ε-SVR)" is then to find such $\widehat{y}$ that has at most ε deviation from the actual output y in feature space, but at the same time is as *flat* as possible. To estimate parameters α_r for $r = 0, \ldots, R$, we thus minimize:

$$E(\alpha) = \sum_{m=1}^{M} |y_m - \widehat{y}_m|_\varepsilon + \lambda \cdot \frac{\|\boldsymbol{\alpha}\|^2}{2}, \tag{A.76}$$

where $|\cdot|_\varepsilon$ is the so-called ε-insensitive loss function:

$$|\xi|_\varepsilon \stackrel{\text{def}}{=} \max\{0,\, |\xi| - \varepsilon\} = \begin{cases} 0 & \text{if } |\xi| \le \varepsilon \\ |\xi| - \varepsilon & \text{otherwise.} \end{cases} \tag{A.77}$$

Note the similarity of the formulations between SVR (Eqs.(A.75) and (A.76)) and artificial neural networks (Eqs.(A.68) and (A.69)), and that *flatness* is ensured by minimizing $\|\boldsymbol{\alpha}\|^2$, equivalent to maximizing the "margin" $\sim 1/\|\boldsymbol{\alpha}\|$. Also note that Eq.(A.76) has the form: "loss + regularizer" as in Eq.(A.52) on page 154 and can be understood from the regularization viewpoint (Smola et al., 1998), and that the loss function is not necessarily ε-insensitive but can be chosen arbitrarily, with the corresponding density models from a probabilistic viewpoint (Section A.2.2).

The beauty of the support vector methods is to solve the "dual" problem—instead of directly searching the solution of the "primal" problem as in Eq.(A.76)—by using the Lagrange multiplier; i.e.,

$$\begin{aligned} &\underset{\boldsymbol{\beta}^+,\boldsymbol{\beta}^-}{\arg\min} \; -\frac{1}{2}\big(\boldsymbol{\beta}^+ - \boldsymbol{\beta}^-\big)^\top \mathbf{K}\big(\boldsymbol{\beta}^+ - \boldsymbol{\beta}^-\big) - \varepsilon \sum_{m=1}^{M}\big(\beta_m^+ + \beta_m^-\big) + \sum_{m=1}^{M}\big(\beta_m^+ - \beta_m^-\big) y_m, \\ &\text{subject to } \sum_{m=1}^{M}\big(\beta_m^+ - \beta_m^-\big) = 0, \quad \text{and} \quad 0 \le \beta_m^+,\, \beta_m^- \le 1/\lambda, \end{aligned} \tag{A.78}$$

where the (i,j)-element of the kernel matrix $\mathbf{K}$ is the inner product (between $\mathbf{x}_i$ and $\mathbf{x}_j$) in feature space: $K_{ij}\big(\mathbf{x}_i, \mathbf{x}_j\big) = \sum_r f_r(\mathbf{x}_i)\, f_r(\mathbf{x}_j)$. The dual problem in Eq.(A.78) can be solved by quadratic programming methods, and the regression estimate then takes the form:

$$\widehat{y} = \alpha_0 + \sum_{m=1}^{M} (\beta_m^+ - \beta_m^-)\, K(\mathbf{x}, \mathbf{x}_m). \tag{A.75'}$$

Therefore, we do not have to specify and work on the large set of (nonlinear) functions f_r, but only the inner product kernel K_{ij} needs to be evaluated ("kernel trick;" Mercer, 1909).

Common choices of the kernels include Gaussian: $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2)$ with $\gamma > 0$—resulting in equivalent model structure as radial basis function (RBF) network—and sigmoidals: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(-\gamma \mathbf{x}_i^\top \mathbf{x}_j + \gamma_0)$, with positive γ, resulting in multilayer perceptrons. From the neural network viewpoint, only those weights from hidden to output layers are modifiable in SVR, and the connection pattern from input to hidden layers is auto-

matically determined, depending on the choice of the kernels. Nevertheless, SVR generally has high expressive power because of the nonlinear projection of inputs—specified by the choice of the kernel—to a high-dimensional feature space (i.e., with a large number of hidden units or support vectors), while at the same time it avoids the "curse of dimensionality" (i.e., the loss of generality due to having a high-dimensional input or feature space; Bellman, 1961) and maintains a high generalization performance by selecting as small number of support vectors as possible.

An appropriate choice of loss functions and regularizers is in most cases critical for the application of SVR in practice. A new parameter ν can be introduced to choose ε and effectively control the fraction of support vectors and errors (ν-SVR; Schölkopf et al., 2000)—by imposing an additional cost: $\nu\varepsilon$ on the primal problem in Eq.(A.76), or constraint: $\sum_{m=1}^{M}\left(\beta_m^+ + \beta_m^-\right) \leq \nu/\lambda$ on the dual problem in Eq.(A.78)—but there is no "free lunch" for these parameter selection procedures, so serendipity and trial-and-error tuning are often required.

A.4 Information Theory

This section overviews the basics and logics of information theory, a powerful tool often used for examining the quality of stimulus-response models in neuroscience (for review, see e.g., Cover and Thomas, 1991; Rieke et al., 1997; Borst and Theunissen, 1999; Paninski, 2003b). I will discuss the connection between correlation functions and entropy/information in Section A.4.1, and a way to compute information by exploiting the SVD and the Fourier transform in Section A.4.2. Finally, Section A.4.3 shows a little survey on entropy estimation by exploiting file compression methods.

A.4.1 Entropy

Entropy is a measure of the randomness—or unpredictability—of observed variables, or the degree of information that the observation gives on the variables (Shannon, 1948). The entropy H of a discrete set of symbols $z = \{z_1, \ldots, z_M\}$ with associated probabilities p_m (for $m = 1, \ldots, M$) is defined as:

$$H(z) \stackrel{\text{def}}{=} -\sum_m p_m \log_2 p_m = \mathbb{E}\big[-\log_2 p_m\big] \quad \text{bits.} \tag{A.79}$$

Similarly, the differential entropy for M continuous variables or vector $\mathbf{x} \in \mathbb{R}^M$ with (Riemann integrable) density $p(\mathbf{x})$ is defined as:

$$H(\mathbf{x}) \stackrel{\text{def}}{=} -\int p(\mathbf{x}) \log_2 p(\mathbf{x})\, d\mathbf{x} = \mathbb{E}\big[-\log_2 p(\mathbf{x})\big] \quad \text{bits.} \tag{A.80}$$

Note the relation between Eqs.(A.79) and (A.80). Let us consider the case with $M = 1$ in Eq.(A.80) for simplicity, where $x \in \mathbb{R}$ is a random variable with density $p(x)$. With the bin size of Δ, the middle value of each bin x_i approximately has the probability: $p_\Delta(x_i) = p(x_i)\,\Delta$. Thus the entropy of the quantized random variable $x_\Delta = \{x_i\}$ is given as, from Eq.(A.79),

$$H(x_\Delta) = -\sum_{i=-\infty}^{\infty} p_\Delta(x_i) \log_2 p_\Delta(x_i) \;\simeq\; -\log_2 \Delta - \sum_{i=-\infty}^{\infty} p(x_i)\,\Delta \log_2 p(z_i) \quad \text{bits.} \tag{A.81}$$

From Eqs.(A.80) and (A.81), we then have:

$$H(x_\Delta) + \log_2 \Delta \;\longrightarrow\; H(x), \quad \text{as} \quad \Delta \;\longrightarrow\; 0. \tag{A.82}$$

Also note that, given the number of symbols M, the maximum entropy discrete distribution is the uniform distribution: $H = \log_2 M$ bits from Eq.(A.79). For continuous distribution, it is instead the Gaussian distribution $\mathbf{x}_{\text{gauss}} \in \mathbb{R}^M$ that has the maximum entropy given covariance $\mathbf{\Lambda}$

(for proof, see e.g., Cover and Thomas, 1991; Rieke et al., 1997):

$$H(\mathbf{x}_{\text{gauss}}) = \log_2 \sqrt{(2\pi e)^M |\det \mathbf{\Lambda}|} \quad \text{bits.} \tag{A.83}$$

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

$$\mathbf{x} = \begin{pmatrix} x(1) \\ x(2) \\ \vdots \\ x(T) \end{pmatrix}, \quad \mathbf{\Lambda} = \mathbb{E}\left[\mathbf{x}\mathbf{x}^\top\right] = \begin{pmatrix} C(0) & C(1) & \cdots & C(T-1) \\ C(-1) & C(0) & \cdots & C(T-2) \\ \vdots & \vdots & \ddots & \vdots \\ C(1-T) & C(2-T) & \cdots & C(0) \end{pmatrix} \tag{A.84}$$

where $C(\tau)$ is the auto-correlation of $x(t)$:

$$C(\tau) \stackrel{\text{def}}{=} \lim_{T\to\infty} \frac{1}{T - |\tau|} \sum_{t=1}^{T} x(t)\, x(t-\tau). \tag{A.85}$$

Note that $C(\tau) = C(-\tau)$ and thus the Toeplitz matrix $\mathbf{\Lambda}$ in Eq.(A.84) is in fact symmetric.

In the case of multiple Gaussian signals $x_m(t)$ for $m = 1, \ldots, M$, we can replace $\mathbf{\Lambda}$ in Eq.(A.83) with the following $M \times M$ block matrix:

$$\mathbf{\Lambda} = \begin{pmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} & \cdots & \mathbf{C}_{1M} \\ \mathbf{C}_{21} & \mathbf{C}_{22} & \cdots & \mathbf{C}_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_{M1} & \mathbf{C}_{M2} & \cdots & \mathbf{C}_{MM} \end{pmatrix} \tag{A.86}$$

where $\mathbf{C}_{ij}$ is the $T \times T$ cross-correlation matrix—an analog of $\mathbf{\Lambda}$ in Eq.(A.84)—between i-th signal $x_i(t)$ and j-th signal $x_j(t)$. Note that $\mathbf{C}_{ij}^\top = \mathbf{C}_{ji}$ and thus the block matrix $\mathbf{\Lambda}$ in Eq.(A.86)

is symmetric. Alternatively, we can first look at between-set covariances at time τ:

$$\mathbf{C}(\tau) = \begin{pmatrix} C_{11}(\tau) & C_{12}(\tau) & \cdots & C_{1M}(\tau) \\ C_{21}(\tau) & C_{22}(\tau) & \cdots & C_{2M}(\tau) \\ \vdots & \vdots & \ddots & \vdots \\ C_{M1}(\tau) & C_{M2}(\tau) & \cdots & C_{MM}(\tau) \end{pmatrix}, \tag{A.87}$$

where $C_{ij}(\tau)$ is the cross-correlation—an analog of $C(\tau)$ in Eq.(A.85)—between $x_i(t)$ and $x_j(t)$:

$$C_{ij}(\tau) \stackrel{\text{def}}{=} \lim_{T\to\infty} \frac{1}{T-|\tau|} \sum_{t=1}^{T} x_i(t)\, x_j(t-\tau). \tag{A.88}$$

We have the covariance matrix $\mathbf{\Lambda}$ as the following $T \times T$ block matrix:

$$\mathbf{\Lambda} = \begin{pmatrix} \mathbf{C}(0) & \mathbf{C}(1) & \cdots & \mathbf{C}(T-1) \\ \mathbf{C}(-1) & \mathbf{C}(0) & \cdots & \mathbf{C}(T-2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}(1-T) & \mathbf{C}(2-T) & \cdots & \mathbf{C}(0) \end{pmatrix}. \tag{A.89}$$

Note the similarity to Eq.(A.84), and that $\mathbf{\Lambda}$ in Eq.(A.89) is symmetric because $\mathbf{C}(\tau) = \mathbf{C}(-\tau)$.

A.4.2 Mutual Information

While entropy measures uncertainty, "information" is defined as the difference of entropies, i.e., a reduction of uncertainty (Shannon, 1948; Cover and Thomas, 1991). 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:

$$I(X,Y) \stackrel{\text{def}}{=} H(X) - H(X|Y), \tag{A.90}$$

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. Because mutual information is symmetric between X and Y, we can also define $I(X,Y) = I(Y,X)$ as:

$$I(X,Y) \stackrel{\text{def}}{=} H(X) + H(Y) - H(X,Y) = H(Y) - H(Y|X), \qquad \text{(A.90')}$$

where $H(X,Y)$ is the joint entropy of X and Y. In the latter expression in Eq.(A.90'), 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.

Estimation of mutual information is often very difficult because it is hard to measure the entropy of experimental data in practice. As a compromise, we often estimate its lower- and/or upper-bound by introducing some assumptions on the distribution (e.g., Gaussianity), and below I will explain some estimation methods commonly used in neuroscience (for details, see e.g., Rieke et al., 1997; Borst and Theunissen, 1999; Paninski, 2003b).

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. 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. We can then calculate $I(Y, Y_{\text{det}})$, which gives an estimated upper-bound of $I(Y,X)$ if we further assume that Y is also Gaussian.

Lower-bound estimate of mutual information

From the data processing inequality theorem, we have: $I(Y,X) \geq I(Y,\widehat{Y})$, where $\widehat{Y}$ is the estimated output of Y from inputs X. If we define: $I_{\text{gauss}} = H(Y) - H(N_{\text{gauss}})$, where N_{gauss} is the Gaussian process with the same dimension and covariance as the estimated noise $N =$

$Y - \widehat{Y}$, then $I(Y, \widehat{Y})$ is bounded below by:

$$I(Y, \widehat{Y}) = H(Y) - H(Y|\widehat{Y}) = H(Y) - H(N) \geq H(Y) - H(N_{\text{gauss}}) = I_{\text{gauss}}. \quad \text{(A.91)}$$

The inequality holds because the Gaussian distribution has the maximum entropy given the covariance. From Eq.(A.83), under the Gaussian assumption on Y, an estimate of mutual information is given as:

$$I_{\text{gauss}} = \frac{1}{2} \log_2 \frac{|\det \mathbf{\Lambda}_Y|}{|\det \mathbf{\Lambda}_N|} \quad \text{(A.92)}$$

where $\mathbf{\Lambda}_Y$ and $\mathbf{\Lambda}_N$ are the covariance matrices of the output Y and the noise N, respectively. Note that I_{gauss} gives the lower-bound of $I(Y, X)$ if and only if Y is in fact Gaussian.

Computation of information using SVD

The most straightforward way to evaluate Eq.(A.92) is to measure the covariance matrices (as in Eq.(A.84) for a single channel, or in Eq.(A.86) or Eq.(A.89) for multiple channels) and to compute their eigenvalues, because $|\det \mathbf{\Lambda}| = \prod_i \lambda_i$, where $\lambda_i \in \mathbb{R}$ are the eigenvalues of $\mathbf{\Lambda}$. However, there are two major difficulties in directly computing $|\det \mathbf{\Lambda}|$ for large data sets: (1) it can be computationally expensive, and (2) it can require huge memory resources. One way to avoid these issues is to introduce a window to approximate the covariance matrices and exploit SVD (see Section Section A.1.2) to compute their eigenvalues.

Consider a single Gaussian signal $x(t)$ for $t = 1, \ldots, T$ whose covariance matrix $\mathbf{\Lambda}$ is given as in Eq.(A.84). Then the symmetric matrix $\mathbf{\Lambda}$ can be approximated as:

$$\mathbf{\Lambda} \approx \mathbf{X}\mathbf{X}^\top, \text{ where } \mathbf{X} = \frac{1}{\sqrt{T}} \begin{pmatrix} x(1) & x(2) & \cdots & x(T) & & & 0 \\ & x(1) & \cdots & x(T-1) & x(T) & & \\ & & \ddots & \vdots & \vdots & \ddots & \\ 0 & & & x(1) & x(2) & \cdots & x(T) \end{pmatrix}. \quad \text{(A.93)}$$

By applying SVD to $\mathbf{X}$, we have the spectral decomposition of $\mathbf{X}\mathbf{X}^\top$:

$$\mathbf{X} = \mathbf{V}\mathbf{\Sigma}\mathbf{U}^\top, \quad \text{and thus} \quad \mathbf{X}\mathbf{X}^\top = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^\top, \tag{A.94}$$

where we follow the notations in Eq.(A.23) on page 140. Although $\mathbf{X}\mathbf{X}^\top$ in Eq.(A.93) is a *biased* estimate of $\mathbf{\Lambda}$ for the off-diagonals, we can then easily compute the determinant of the covariance matrix $\mathbf{\Lambda}$ as:

$$|\det \mathbf{\Lambda}| = \left|\det \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^\top\right| = \prod_i \sigma_i^2, \tag{A.95}$$

where σ_i are the singular values of $\mathbf{X}$ and σ_i^2 correspond to the eigenvalues of $\mathbf{\Lambda}$. In the case of multiple Gaussian signals $x_m(t)$ for $m = 1, \ldots, M$, we can evaluate Eq.(A.92) in a similar manner. That is, an $\mathbf{X}$ that satisfies $\mathbf{\Lambda} \approx \mathbf{X}\mathbf{X}^\top$ is given as the following block matrix:

$$\mathbf{X} = \begin{pmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_M \end{pmatrix} \tag{A.96}$$

where $\mathbf{X}_i$ is the analog of $\mathbf{X}$ in Eq.(A.93) for the i-th signal $x_i(t)$.

Although an efficient algorithm for the SVD has been provided elsewhere (e.g., svd function—with the "econ" option—in MATLAB), it would not necessarily be computationally preferable to apply SVD directly to $\mathbf{X}$ because the $TM \times (2T-1)$ matrix $\mathbf{X}$ can be bigger than the $TM \times TM$ covariance matrix $\mathbf{\Lambda}$ 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 $K\,(\leq T)$ to approximate the covariance matrix $\mathbf{\Lambda} \approx \bar{\mathbf{X}}\bar{\mathbf{X}}^\top$, where $\bar{\mathbf{X}}$ is the $K \times (T + K - 1)$ matrix corresponding to the upper-left corner—the first K rows of $\mathbf{X}$ in Eq.(A.93) in essence—of $\mathbf{X}$ in Eq.(A.93) in the single channel case; in the multiple channel case, we have the $KM \times (T+K-1)$ matrix $\bar{\mathbf{X}}$ as an analog of Eq.(A.96). Note that having the window length K results in the same approximation level as having the bin size $2\pi K/T$ for the analysis in the Fourier domain (see below).

Furthermore, we can (randomly) pick up $L\,(\leq T-K+1)$ samples (columns) to obtain an $\bar{\mathbf{X}}$ analog, resulting in the $K\times L$ and $KM\times L$ matrices in the single and multiple channel case, respectively. In the former case, we have:

$$\frac{1}{\sqrt{L}}\begin{pmatrix} x(i_1+K-1) & x(i_2+K-1) & \cdots & x(i_L+K-1) \\ x(i_1+K-2) & x(i_2+K-2) & \cdots & x(i_L+K-2) \\ \vdots & \vdots & & \vdots \\ x(i_1) & x(i_2) & \cdots & x(i_L) \end{pmatrix}. \tag{A.97}$$

In this way, we can reasonably approximate the covariance matrices and readily evaluate Eq.(A.92) in the time domain.

Computation of mutual information in the Fourier domain

Although Eq.(A.92) holds in any orthonormal basis, it is in many cases evaluated in the Fourier domain under the assumption of stationary (time translation-invariant) ensembles. The main reason is that the covariance matrices in the Fourier domain are diagonal because the Fourier transform is an 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:

$$\log_2|\det\mathbf{\Lambda}_Y| = \sum_{\omega}\log_2\mathcal{P}_Y(\omega), \quad \log_2|\det\mathbf{\Lambda}_N| = \sum_{\omega}\log_2\mathcal{P}_N(\omega), \tag{A.98}$$

where $\mathcal{P}_Y(\omega)$ and $\mathcal{P}_N(\omega)$ are the power spectral densities of the outputs Y and the noise N, respectively. The power spectral density can be obtained by the squared Fourier coefficients of the signals, or the Fourier transform of the auto-correlation function (that is, for large T in Eq.(A.84), the eigenvalues of $\mathbf{\Lambda}$ correspond to the power spectral density of $\mathbf{x}$; the Wiener-

Khintchine theorem). Then we have, from Eqs.(A.92) and (A.98),

$$I_\omega = \frac{1}{2}\log_2\frac{\mathcal{P}_Y(\omega)}{\mathcal{P}_N(\omega)}, \quad I_{\text{gauss}} = \sum_\omega I_\omega = \sum_{\omega=0}^{f_c}\log_2\frac{\mathcal{P}_Y(\omega)}{\mathcal{P}_N(\omega)} \quad \text{bits} \tag{A.99}$$

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

In the case of multiple dynamic channels, it requires huge computational power and resources to evaluate Eq.(A.92) in the time domain using Eq.(A.86), because we need to consider the correlation between the channels as well. For example, in a linear decoding (reconstruction) model where neural response $r(t)$ is used to estimate input spectrogram $S(t, f)$, the covariance matrix of S has $\mathcal{O}(T^2M^2)$ elements where T is the sample size in time and M is the number of frequency bands in S or the number of channels—we introduced the SVD method to avoid this issue. In contrast, two-dimensional Fourier transform leads to a diagonal covariance matrix of the Fourier coefficients. Therefore, it is much easier to evaluate Eq.(A.92) in the Fourier domains, and the estimate of mutual information $I(S, r)$ under the Gaussian assumption is:

$$I_{\text{gauss}} = \sum_{t,\,m} I(\omega_t,\,\omega_m) = \sum_{t,\,m}\log_2\frac{\mathcal{P}_S(\omega_t,\,\omega_m)}{\mathcal{P}_N(\omega_t,\,\omega_m)} \quad \text{bits}, \tag{A.100}$$

where ω_t and ω_m are the Fourier domains corresponding to time and frequency in the spectrogram, respectively. $I(\omega_t,\,\omega_m)$ is the information at $(\omega_t,\,\omega_m)$, and $\mathcal{P}_S$ and $\mathcal{P}_N$ are the squared Fourier coefficients (power) of the input spectrogram and the reconstruction noise, respectively.

Signal-to-noise ratio and coherence function

Several equivalent formulae for Eq.(A.99) can be derived in the linear system, using the signal-to-noise ratio (SNR) and coherence function (Gabbiani, 1996; Rieke et al., 1997). Let $Y(t)$ and $\widehat{Y}(t)$ be the output and its estimate from inputs $X(t)$, respectively. Then the estimated noise is

given as: $N(t) = Y(t) - \widehat{Y}(t)$, and the SNR is defined as:

$$\text{SNR}(\omega) = \frac{\mathcal{P}_{\widehat{Y}}(\omega)}{\mathcal{P}_N(\omega)} = \frac{\mathcal{P}_Y(\omega)}{\mathcal{P}_N(\omega)} - 1, \tag{A.101}$$

where $\mathcal{P}_Y$, $\mathcal{P}_{\widehat{Y}}$, and $\mathcal{P}_N$ are the power spectral densities of Y, $\widehat{Y}(t)$, and $N(t)$, respectively. The Gaussian estimate of mutual information between X and Y can then be written as, from Eqs.(A.99) and (A.101),

$$I_{\text{gauss}} = \sum_{\omega} \log_2\big[1 + \text{SNR}(\omega)\big] \quad \text{bits.} \tag{A.99'}$$

The SNR can also be defined as:

$$\text{SNR}(\omega) = \frac{\mathcal{P}_Y(\omega)}{\mathcal{P}_{N_{\text{eff}}}(\omega)}, \tag{A.101'}$$

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

$$\widehat{Y}(t) = g(t) * \big(Y(t) + N_{\text{eff}}(t)\big). \tag{A.102}$$

Here $*$ denotes 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 time t. The Fourier transform of $g(t)$ is then called the coherence function $\widetilde{g}(\omega)$, and the Gaussian estimate can be given as:

$$I_{\text{gauss}} = -\sum_{\omega} \log_2\big[1 - \widetilde{g}(\omega)\big] \quad \text{bits.} \tag{A.99"}$$

In a linear system, the coherence function between the inputs $X(t)$ and the outputs $Y(t)$ can be rewritten as:

$$\widetilde{g}(\omega) = \frac{|\mathcal{P}_{XY}(\omega)|^2}{\mathcal{P}_X(\omega)\mathcal{P}_Y(\omega)} = \frac{\text{SNR}(\omega)}{1 + \text{SNR}(\omega)}, \tag{A.103}$$

where $\mathcal{P}_{XY}$ is the Fourier transform of the cross-correlation between $X(t)$ and $Y(t)$, and $\mathcal{P}_X$ and $\mathcal{P}_Y$ are the power spectral densities of $X(t)$ and $Y(t)$, respectively.

A.4.3 File Compression Methods

Entropy (mutual information) estimation has been widely investigated, and various methods have been proposed in the context of neuroscience: e.g., for estimating the entropy carried by a spike train (e.g., histogram method, Strong et al., 1998; vector space method, Victor, 2002; Lempel-Ziv complexity method, Amigó et al., 2004). The methods described above have also been widely used, but the Gaussian assumption does not hold in most cases and thus the estimates could be far different from the real values. Considering that the entropy is in some sense the coding length of random variables, i.e., the number of bits required to encode the data (the "source coding" theorem; Cover and Thomas, 1991), we could instead estimate the entropy of data by examining how well the data can be compressed.

Say that each sample point in raw data is encoded by K bits/sample. Then the original file size for a given data of length M is $X_{\text{orig}} = MK$ bits. Suppose the actual entropy of the data is H bits/sample. Based on the source coding theorem, the maximally compressed file size for the data (X_{comp} bits) would then be asymptotically equivalent to HM bits. Thus the estimated entropy can be given as:

$$\widehat{H} = K \frac{X_{\text{comp}}}{X_{\text{orig}}} = \frac{X_{\text{comp}}}{M} \ (\geq H) \quad \text{bits/sample}. \tag{A.104}$$

For better entropy estimation, X_{comp} can be normalized by subtracting the compressed file size for zero entropy data of length M.

In practice, there are a couple of points we should be aware of. First, there is no *best* compression algorithm in general and thus the entropy based on Eq.(A.104) would give overestimates. Nevertheless, the compression approach surely has an advantage because it takes temporal structures of data into account more directly than discretization approaches such as the direct method.

The second point is that higher data precision requires larger data size. To avoid this issue, floating-point data would be converted into integers by multiplying data with a constant

α/β, which gives data precision of β/α with the data range of $[-\beta,\ \beta]$. Note however that this quantization process is *lossy*.

File compression algorithms

To encode a given data with as small file size as possible, various lossless file compression methods have been developed for the past decades, such as the Run-Length method and the Huffman coding method (Huffman, 1952; for review, see e.g., Salomon, 2006). These methods are typically based on Shannon's information entropy (Shannon, 1948), and here we examined the performance of several commercially available compressors. The gzip algorithm,[5] which is widely used in the UNIX and Windows platforms, is based on the Deflate/Inflate compression algorithm—Lempel-Ziv algorithm (Ziv and Lempel, 1977, 1978) with a sliding dictionary method and the Huffman coding—and often shows a good performance. The bzip2 algorithm[6] uses the Burrows-Wheeler transform (BWT) algorithm (Burrows and Wheeler, 1994) and Huffman coding. The bzip2 algorithm often outperforms the more conventional Lempel-Ziv (LZ77/LZ88 or LZW; Welch, 1984) algorithms, and approaches the performance of the prediction-by-partial-matching (PPM) family of statistical methods (Cleary and Witten, 1984). We also used the rk archiver[7] which often shows a very good compression performance.[8]

Entropy estimation for Gaussian distribution

To examine the performance of entropy estimation by the file compression method, we first applied it to an uncorrelated Gaussian distribution (white noise; Figure A.4A–B) with various variances as well as correlated Gaussian distributions (pink noise; Figure A.4C–E). Note that the entropy of a Gaussian distribution is given by Eq.(A.83) on page 168, as the sum of entropies for each Fourier component (see Eq.(A.98) on page 173). As is expected, the obtained

[5]The latest distribution gzip 1.2.4 (as of July, 2007): http://www.gzip.org/
[6]The latest distribution bzip2 1.0.4 (as of July, 2007): http://sources.redhat.com/bzip2/
[7]The latest distribution rk 1.04.1 (as of July, 2007): http://www.msoftware.co.nz/
[8]For benchmark, see e.g., http://www.maximumcompression.com/

values overestimated the true entropy in all the tested examples, but bzip2 and rk performed better than gzip. The estimated entropy as in Eq.(A.104) by the three algorithms depended little on the data length (tested data length ranged from 1×10^4 up to 1×10^7; data not shown).

Entropy estimation for natural sounds

We then examined the performance of entropy estimation for natural sound fragments. All fragments were obtained from commercially available audio CDs, originally sampled at 44.1 kHz and resampled at 97.656 kHz; *The Diversity of Animal Sounds* and *Sounds of Neotropical Rainforest Mammals* (Cornell Laboratory of Ornithology, Ithaca, NY, USA). For comparison, the upper-bound entropy was also estimated as the sum of entropies for each Fourier component under the Gaussian assumption. Here we multiplied the data in 16-bit encodings by $\alpha/\beta = 1$, 10, 100, $2^{15}/100$, and $2^{16}/100$, and rounded to the nearest integers for quantization.

As shown in Figure A.4F, the estimated entropies by bzip2 and rk were consistently smaller than the upper-bound estimates calculated under the Gaussian assumption, whereas those by gzip were sometimes not. Considering that the file compression methods return somewhat overestimates, the estimated entropies should be closer to the real values than those obtained under the Gaussian assumption. This makes sense because natural sounds presumably have rich temporal and spectral structures and differ much from Gaussian distributions.

These results suggest some advantages of the file compression method where we have no assumption on the data structures. Because the compression method considers the temporal structure of data more directly than discretization approaches, it would potentially give a good estimate for natural sounds or natural images. Another advantage would be the ease of estimation. Since the performance depends on the compression algorithm, however, the entropy estimation would be slightly biased and often overestimated. It should also be noted that the file compression methods typically require a large data set to estimate the entropy, and that our survey here illustrates well the difficulties of estimating entropy in general.

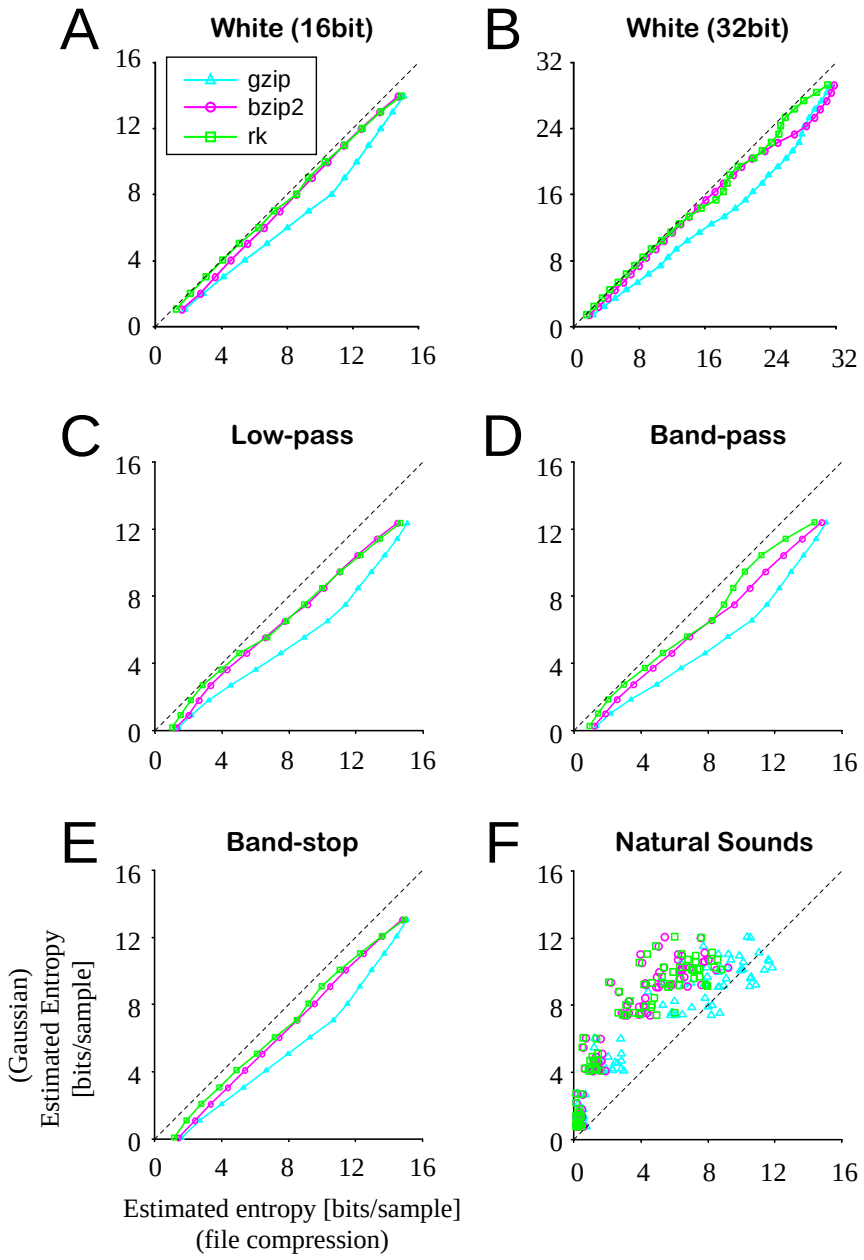

Figure A.4: **Entropy estimation by file compression**. The compression level was chosen to be the best (switches: -9 for gzip and bzip2, and -mx for rk). (**A–B**) White noise of the data length 1×10^7 in 16 and 32 bit encodings. Relatively lower performances around every 8 bits (i.e., 1 byte) would be due to the specification of the file system. (**C–E**) Pink noise (data length; 1×10^7) in 16 bit encodings. The rk (*green*) and bzip2 (*magenta*) performed better than the gzip (*cyan*) for estimating entropy in all examples we tested; low-pass, high-pass (data not shown, but the results were very similar to those of low-pass filtered case), band-pass, and band-stop filtered noise. (**F**) Natural sounds (16 bit encodings; 16 seconds; sampling frequency, 97.656 kHz) with quantization factor $\alpha/\beta = 1$, 10, 100, $2^{15}/100$, and $2^{16}/100$. In most cases rk gave the best (i.e., the smallest) estimates.