

Learning Overcomplete Representations

Michael S. Lewicki

Computer Science Dept. and Center for the Neural Basis of Cognition, Carnegie Mellon Univ., 115 Mellon Inst., 4400 Fifth Ave., Pittsburgh, PA 15213

Terrence J. Sejnowski

Howard Hughes Medical Institute, Computational Neurobiology Laboratory, The Salk Institute, La Jolla, CA 92037, U.S.A.

In an overcomplete basis, the number of basis vectors is greater than the dimensionality of the input, and the representation of an input is not a unique combination of basis vectors. Overcomplete representations have been advocated because they have greater robustness in the presence of noise, can be sparser, and can have greater flexibility in matching structure in the data. Overcomplete codes have also been proposed as a model of some of the response properties of neurons in primary visual cortex. Previous work has focused on finding the best representation of a signal using a fixed overcomplete basis (or dictionary). We present an algorithm for learning an overcomplete basis by viewing it as probabilistic model of the observed data. We show that overcomplete bases can yield a better approximation of the underlying statistical distribution of the data and can thus lead to greater coding efficiency. This can be viewed as a generalization of the technique of independent component analysis and provides a method for Bayesian reconstruction of signals in the presence of noise and for blind source separation when there are more sources than mixtures.

1 Introduction ---

A common way to represent real-valued signals is with a linear superposition of basis functions. This can be an efficient way to encode a high-dimensional data space because the representation is distributed. Bases such as the Fourier or wavelet can provide a useful representation of some signals, but they are limited because they are not specialized for the signals under consideration.

An alternative and potentially more general method of signal representation uses so-called overcomplete bases (also called overcomplete dictionaries), which allow a greater number of basis functions (also called dictionary elements) than samples in the input signal (Simoncelli, Freeman, Adelson, & Heeger, 1992; Mallat & Zhang, 1993; Chen, Donoho, & Saunders, 1996). Overcomplete bases are typically constructed by merging a set of complete

bases (e.g., Fourier, wavelet, and Gabor), or by adding basis functions to a complete basis (e.g., adding frequencies to a Fourier basis).

Under an overcomplete basis, the decomposition of a signal is not unique, but this can offer some advantages. One is that there is greater flexibility in capturing structure in the data. Instead of a small set of general basis functions, there is a larger set of more specialized basis functions such that relatively few are required to represent any particular signal. These can form more compact representations, because each basis function can describe a significant amount of structure in the data. For example, if a signal is largely sinusoidal, it will have a compact representation in a Fourier basis. Similarly, a signal composed of chirps is naturally represented in a chirp basis. Combining both of these bases into a single overcomplete basis would allow compact representations for both types of signals (Coifman & Wickerhauser, 1992; Mallat & Zhang, 1993; Chen et al., 1996). It is also possible to obtain compact representations when the overcomplete basis contains a single class of basis functions. An overcomplete Fourier basis, with more than the minimum number of sinusoids, can compactly represent signals composed of small numbers of frequencies, achieving superresolution (Chen et al., 1996). An additional advantage of some overcomplete representations is increased stability of the representation in response to small perturbations of the signal (Simoncelli et al., 1992).

Unlike the case in a complete basis, where signal decomposition is well defined and unique, finding the "best" representation in terms of an overcomplete basis is a challenging problem. It requires both an objective for decomposition and an algorithm that can achieve that objective. Decomposition can be expressed as finding a solution to

$$\mathbf{x} = \mathbf{A}\mathbf{s}, \quad (1.1)$$

where \mathbf{x} is the signal, \mathbf{A} is a (nonsquare) matrix of basis functions (vectors), and \mathbf{s} is the vector of coefficients, that is, the representation of the signal. Developing efficient algorithms to solve this equation is an active area of research. One approach to removing the degeneracy in equation 1.1 is to place a constraint on \mathbf{s} (Daubechies, 1990; Chen et al., 1996), for example, by finding \mathbf{s} satisfying equation 1.1 with minimum L_1 norm. A different approach is to construct iteratively a sparse representation of the signal (Coifman & Wickerhauser, 1992; Mallat & Zhang, 1993). In some of these approaches, the decomposition can be a nonlinear function of the data.

Although overcomplete bases can be more flexible in terms of how the signal is represented, there is no guarantee that hand-selected basis vectors will be well matched to the structure in the data. Ideally, we would like the basis itself to be adapted to the data, so that for signal class of interest, each basis function captures a maximal amount of structure.

One recent success along these lines was developed by Olshausen and Field (1996, 1997) from the viewpoint of learning sparse codes. When adapted

to natural images, the basis functions shared many properties with neurons in primary visual cortex, suggesting that overcomplete representations might a useful model for neural population codes.

In this article, we present an algorithm for learning an overcomplete basis by viewing it as a probabilistic model of the observed data. This approach provides a natural solution to decomposition (see equation 1.1) by finding the maximum a posteriori representation of the data. The prior distribution on the basis function coefficients removes the redundancy in the representation and leads to representations that are sparse and are a nonlinear function of the data. The probabilistic approach to decomposition also leads to a natural method of denoising.

From this model, we derive a simple and robust learning algorithm by maximizing the data likelihood over the basis functions. This work generalizes the algorithm of Olshausen and Field (1996) by deriving an algorithm for learning overcomplete bases from a direct approximation to the data likelihood. This allows learning for arbitrary input noise levels, allows for the objective comparison of different models, and provides a way to estimate a model's coding efficiency. We also show that overcomplete representations can provide a better and more efficient representation, because they can better approximate the underlying statistical density of the input data. This also generalizes the technique of independent component analysis (Jutten & Héroult, 1991; Comon, 1994; Bell & Sejnowski, 1995) and provides a method for the identification of more sources than mixtures.

2 Model

We assume that each data vector, $\mathbf{x} = x_1, \dots, x_L$, can be described with an overcomplete linear basis plus additive noise,

$$\mathbf{x} = \mathbf{A}\mathbf{s} + \boldsymbol{\epsilon}, \quad (2.1)$$

where \mathbf{A} is an $L \times M$ matrix with $M > L$. We assume gaussian additive noise, $\boldsymbol{\epsilon}$. The data likelihood is

$$\log P(\mathbf{x}|\mathbf{A}, \mathbf{s}) \propto -\frac{1}{2\sigma^2}(\mathbf{x} - \mathbf{A}\mathbf{s})^2, \quad (2.2)$$

where σ^2 is the noise variance.

One criticism of overcomplete representations is that they are redundant—a given data point can have many possible representations—but this redundancy can be removed by a proper choice for the prior probability of the basis coefficients, $P(\mathbf{s})$, which specifies the probability of the alternative representations. This density determines how the underlying statistical structure is modeled and the nature of the representation. Standard approaches to signal representation do not specify a prior for the coefficients, because

for most complete bases and assuming zero noise, the representation of the signal is unique. If \mathbf{A} is invertible, the decomposition of the signal \mathbf{x} is given by $\mathbf{s} = \mathbf{A}^{-1}\mathbf{x}$. Because \mathbf{A}^{-1} is expensive to compute, there is strong incentive to find basis matrices that are easy to invert, such as restricting the basis functions to be orthogonal or further restricting the basis functions to those for which there are fast algorithms, such as Fourier or wavelet analysis.

A more general approach is afforded in the probabilistic formulation. The parameter values of the internal states are found by maximizing the posterior distribution of \mathbf{s} ,

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\operatorname{argmax}} P(\mathbf{s}|\mathbf{x}, \mathbf{A}) = \underset{\mathbf{s}}{\operatorname{argmax}} P(\mathbf{x}|\mathbf{A}, \mathbf{s})P(\mathbf{s}), \quad (2.3)$$

where $\hat{\mathbf{s}}$ is the most probable decomposition of the signal. This formulation of the problem offers the advantage that the model can fit more general types of distributions. For simplicity, we assume independence of the coefficients: $P(\mathbf{s}) = \prod_m P(s_m)$. Another advantage of this formulation is that the process of finding the most probable representation determines the best representation for the noise level defined by σ . This automatically performs “denoising” in that \mathbf{s} encodes the underlying signal without noise. The noise level can be set to arbitrary levels, including zero. It is also possible to use Bayesian methods to infer the most probable noise level, but this will not be pursued here.

2.1 The Relation Between the Prior and the Representation. Figure 1 shows how different priors induce different representations of the data. A standard choice might be a gaussian prior (equivalent to factor analysis), but this yields no advantage to having an overcomplete representation because the underlying assumption is still that the data are gaussian. An alternative choice, advocated by some authors (Field, 1994; Olshausen & Field, 1996; Chen et al., 1996), is to use priors that assume sparse representations. This is accomplished by priors that have high kurtosis, such as the Laplacian, $P(s_m) \propto \exp(-\theta|s_m|)$. Compared to a gaussian, this distribution puts greater weight on values close to zero, and as result the representations are sparser—they have a greater number of small-valued coefficients. For overcomplete bases, we expect a priori that the representation will be sparse, because only L out of M nonzero coefficients are needed to represent arbitrary L -dimensional input patterns.

In the case of zero noise and $P(\mathbf{s})$ gaussian, maximizing equation 2.3 is equivalent to $\min_{\mathbf{s}} \|\mathbf{s}\|_2$ subject to $\mathbf{x} = \mathbf{A}\mathbf{s}$. The solution can be obtained with the pseudoinverse, $\mathbf{s} = \mathbf{A}^+\mathbf{x}$, and is a linear function of \mathbf{A} and \mathbf{s} . In the case of zero noise and $P(\mathbf{s})$ Laplacian, maximizing equation 2.3 is equivalent to $\min_{\mathbf{s}} \|\mathbf{s}\|_1$ subject to $\mathbf{x} = \mathbf{A}\mathbf{s}$. Unlike the gaussian prior, this solution cannot be obtained by a simple linear operation.

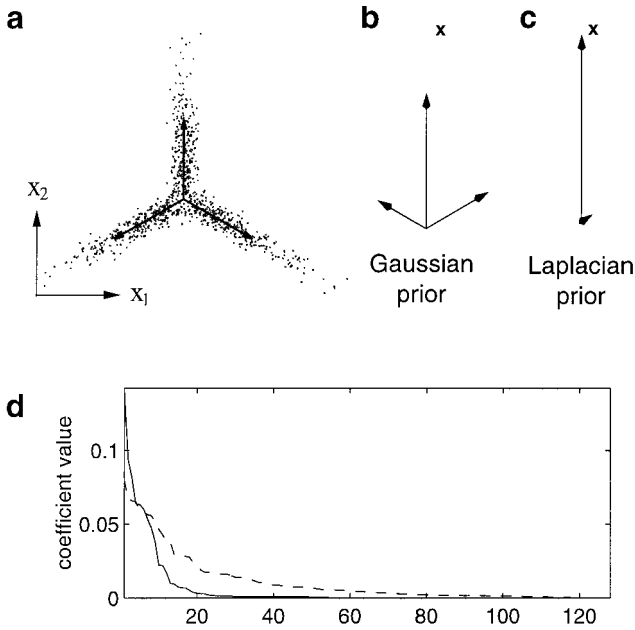


Figure 1: Different priors induce different representations. (a) The data distribution is two-dimensional and has three main arms. The overlaid axes form an overcomplete representation. (b, c) Optimal scaled basis vectors for the data point \mathbf{x} under a gaussian and Laplacian prior, respectively. Assuming low noise, a gaussian for $P(\mathbf{s})$ is equivalent to finding \mathbf{s} with minimum L_2 norm such that $\mathbf{x} = \mathbf{A}\mathbf{s}$. The solution is given by the pseudoinverse $\mathbf{s} = \mathbf{A}^+\mathbf{x}$ and is a linear function of the data. A Laplacian prior ($P(s_m) \propto \exp[-\theta|s_m|]$) finds \mathbf{s} with minimum L_1 norm. This is a nonlinear operation that essentially selects a subset of basis vectors to represent the data (Chen et al., 1996), so that the resulting representation is sparse. (d) A 64-sample segment of natural speech was fit to a $2\times$ overcomplete Fourier representation (128 basis functions) (see section 7). The plot shows rank-order distribution of the coefficients of \mathbf{s} under a gaussian prior (dashed) and a Laplacian prior (solid). Under a gaussian prior, nearly all of the coefficients in the solution are nonzero, whereas far fewer are nonzero under a Laplacian prior.

3 Inferring the Internal State

A general approach for optimizing \mathbf{s} in the case of finite noise ($\epsilon > 0$) and nongaussian $P(\mathbf{s})$ is to use the gradient of the log posterior in an optimization algorithm (Daugman, 1988; Olshausen & Field, 1996). A suitable initial condition is $\mathbf{s} = \mathbf{A}^T\mathbf{x}$ or $\mathbf{s} = \mathbf{A}^+\mathbf{x}$.

An alternative method, which can be used when the prior is Laplacian and $\epsilon = 0$, is to view the problem as a linear program (Chen et al., 1996):

$$\min \mathbf{c}^T |\mathbf{s}| \quad \text{subject to} \quad \mathbf{A}\mathbf{s} = \mathbf{x}. \quad (3.1)$$

Letting $\mathbf{c} = (1, \dots, 1)$, the objective function in the linear program, $\mathbf{c}^T |\mathbf{s}| = \sum_m |s_m|$, corresponds to maximizing the log posterior likelihood under a Laplacian prior. This can be converted to a standard linear program (with only positive coefficients) by separating positive and negative coefficients. Making the substitutions, $\mathbf{s} \leftarrow [\mathbf{u}; \mathbf{v}]$, $\mathbf{c} \leftarrow [1; 1]$, and $\mathbf{A} \leftarrow [\mathbf{A}; -\mathbf{A}]$, equation 3.1 becomes

$$\min \mathbf{1}^T [\mathbf{u}; \mathbf{v}] \quad \text{subject to} \quad [\mathbf{A}; -\mathbf{A}][\mathbf{u}; \mathbf{v}] = \mathbf{x}, \quad \mathbf{u}, \mathbf{v} \geq 0, \quad (3.2)$$

which replaces the basis vector matrix \mathbf{A} with one that contains both positive and negative copies of the vectors. This separates the positive and negative coefficients of the solution \mathbf{s} into the positive variables \mathbf{u} and \mathbf{v} , respectively. This can be solved efficiently and exactly with interior point linear programming methods (Chen et al., 1996). Quadratic programming approaches to this type of problem have also recently been suggested (Osuna, Freund, & Girosi, 1997) for similar problems.

We have used both the linear programming and gradient-based methods. The linear programming methods were superior for finding exact solutions in the case of zero noise. The standard implementation handles only the noiseless case but can be generalized (Chen et al., 1996). We found gradient-based methods to be faster in obtaining good approximate solutions. They also have the advantage that they can easily be adapted for more general models, such as positive noise levels or different priors.

4 Learning

To derive a learning algorithm we must first specify an appropriate objective function. A natural objective is to maximize the probability of the data given the model. For a set of K independent data vectors $\mathbf{X} = \mathbf{x}_1, \dots, \mathbf{x}_K$,

$$P(\mathbf{X}|\mathbf{A}) = \prod_{k=1}^K P(\mathbf{x}_k|\mathbf{A}). \quad (4.1)$$

The probability of a single data point is computed by marginalizing over the states of the network,

$$P(\mathbf{x}_k|\mathbf{A}) = \int d\mathbf{s} P(\mathbf{s}) P(\mathbf{x}_k|\mathbf{A}, \mathbf{s}). \quad (4.2)$$

This formulates the problem as one of density estimation and is equivalent to minimizing the Kullback-Leibler divergence between the model density

and the distribution of the data. If implementation-related issues such as synaptic noise are ignored, this is equivalent to the methods of redundancy reduction and maximizing the mutual information between the input and the representation (Nadal & Parga, 1994a, 1994b; Cardoso, 1997), which have been advocated by several researchers (Barlow, 1961, 1989; Hinton and Sejnowski, 1986; Daugman, 1989; Linsker, 1988; Atick, 1992).

4.1 Fitting Bases to the Data Distribution. To understand how overcomplete representations can yield better approximations to the underlying density, it is helpful to contrast different techniques for adapting a basis to a particular data set.

Principal component analysis (PCA) models the data with a multivariate gaussian. This representation is commonly used to find the directions in the data with largest variation—the principal components—even if the data do not have gaussian structure. The basis functions are the eigenvectors of the covariance matrix and are restricted to be orthogonal. An extension of PCA, called independent component analysis (ICA) (Jutten & Héroult, 1991; Comon, 1994; Bell & Sejnowski, 1995), allows the learning of nonorthogonal bases for data with nongaussian distributions. ICA is highly effective in several applications such as blind source separation of mixed audio signals (Jutten & Héroult, 1991; Bell & Sejnowski, 1995), decomposition of electroencephalographic (EEG) signals (Makeig, Jung, Bell, Ghahremani, & Sejnowski, 1996), and the analysis of functional magnetic resonance imaging (fMRI) data (McKeown et al., 1998). In all of these techniques, the number of basis vectors is equal to the number of inputs. Because these bases span the input space, they are complete and are sufficient to represent the data, but we will see that this representation can be limited.

Figure 2 illustrates how different data densities are modeled by various approaches in a simple two-dimensional data space. PCA assumes gaussian structure, but if the data have nongaussian structure, the vectors can point in directions that contain very few data, and the probability density defined by the model will predict data where none occur. This means that the model will underestimate the likelihood of data in the dense regions and overestimate it in the sparse regions. By Shannon's theorem, this will limit the efficiency of the representation. ICA assumes the coefficients have nongaussian structure and allows the vectors to be nonorthogonal. In this example, the basis vectors point along high-density regions of the data space. If the density is more complicated, however, as in the case of the three-armed density, neither PCA nor ICA captures the underlying structure. Although it is possible to represent every point in the two-dimensional space with a linear combination of two vectors, the underlying density cannot be described without specifying a complicated prior on the basis function coefficients. An overcomplete basis, however, allows an efficient representation of the underlying density with simple priors on the basis function coefficients.

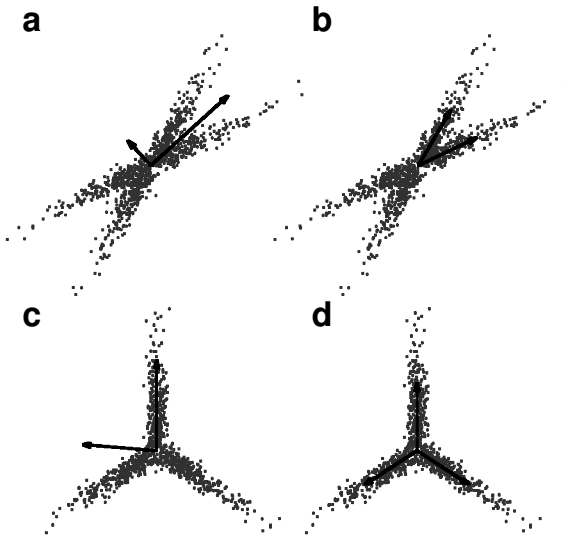


Figure 2: Fitting two-dimensional data with different bases. (a) PCA makes the assumption that the data have a gaussian distribution. The optimal basis vectors are orthogonal and are not efficient at representing nonorthogonal density distributions. (b, c) ICA does not require that the vectors be orthogonal and can fit more general types of densities. (c) Some data distributions, however, cannot be modeled adequately by either PCA or ICA. (d) Allowing the basis to be overcomplete allows the three-armed distribution to be fit with simple and independent distributions for the basis vector coefficients.

4.2 Approximating the Data Probability. In deriving the learning algorithm, the first problem is that, in general, the integral in equation 4.2 is intractable. In the special case of zero noise and a complete representation (i.e., \mathbf{A} is invertible) this integral can be solved and leads to the well-known ICA algorithm (MacKay, 1996; Pearlmutter & Parra, 1997; Olshausen & Field, 1997; Cardoso, 1997). But in the case of an overcomplete basis, such a solution is not possible. Some recent approaches have tried to approximate this integral by evaluating $P(\mathbf{s})P(\mathbf{x}|\mathbf{A}, \mathbf{s})$ at its maximum (Olshausen & Field, 1996, 1997), but this ignores the volume information of the posterior. This means representations that are well determined (i.e., have a sharp posterior distribution) are treated in the same way as those that are ill determined, which can introduce biases into the learning. Another problem is that this leads to a trivial solution where the magnitude of the basis functions diverge. This problem can be somewhat circumvented by adaptive normalization (Olshausen & Field, 1996), but setting the adaptation rates can be tricky in practice, and, more important,

there is no guarantee the desired objective function is the one being optimized.

The approach we take here is to approximate equation 4.2 with a gaussian around the posterior mode, $\hat{\mathbf{s}}$. This yields

$$\begin{aligned} \log P(\mathbf{x}|\mathbf{A}) \approx & \frac{L}{2} \log \frac{\lambda}{2\pi} + \frac{M}{2} \log(2\pi) + \log P(\hat{\mathbf{s}}) - \frac{\lambda}{2} (\mathbf{x} - \mathbf{A}\hat{\mathbf{s}})^2 \\ & - \frac{1}{2} \log \det \mathbf{H}, \end{aligned} \quad (4.3)$$

where $\lambda = 1/\sigma^2$, and \mathbf{H} is the Hessian of the log posterior at $\hat{\mathbf{s}}$, $\mathbf{H} = \lambda \mathbf{A}^T \mathbf{A} - \nabla_{\mathbf{s}} \nabla_{\mathbf{s}} \log P(\hat{\mathbf{s}})$. This is a saddle-point approximation. (See appendix for derivation details.) Care must be taken that the log prior has some curvature, because $\det(\mathbf{A}^T \mathbf{A}) = 0$ arising from the fact that $\mathbf{A}^T \mathbf{A}$ has the same rank as \mathbf{A} , which is assumed to be rectangular. The accuracy of the saddle-point approximation is determined by how closely the posterior mode can be approximated by a gaussian. The approximation will be poor if there are multiple modes or there is excessive skew or kurtosis. We present methods for addressing some of these issues in section 6.1.

A learning rule is obtained by differentiating $\log P(\mathbf{x}|\mathbf{A})$ with respect to \mathbf{A} (see the appendix), which leads to the following expression:

$$\Delta \mathbf{A} = \mathbf{A} \mathbf{A}^T \nabla_{\mathbf{A}} \log P(\mathbf{x}|\mathbf{A}) \approx -\mathbf{A}(\mathbf{z}\hat{\mathbf{s}}^T + \mathbf{I}), \quad (4.4)$$

where $z_k = \partial \log P(s_k) / \partial s_k$. This rule contains no matrix inverses, and the vector \mathbf{z} involves only the derivative of the log prior.

In the case where \mathbf{A} is square, this form of the rule is exactly the natural gradient ICA learning rule for the basis matrix (Amari, Cichocki, & Yang, 1996). The difference in the more general case where \mathbf{A} is rectangular is in how the coefficients \mathbf{s} are calculated. In the standard ICA learning algorithm (\mathbf{A} square, zero noise), the coefficients are given by $\mathbf{s} = \mathbf{W}\mathbf{x}$, where $\mathbf{W} = \mathbf{A}^{-1}$ is the filter matrix. For the learning rule to work the optimization of \mathbf{s} must maximize the posterior distribution $P(\mathbf{s}|\mathbf{x}, \mathbf{A})$ (see equation 2.3).

5 Examples

We now demonstrate the algorithm on two simple data sets. Figure 3 shows the results of fitting two different two-dimensional data sets. The learning procedure was as follows. The initial bases were random normalized vectors with the constraint that no two vectors be closer in angle than 30 degrees. Each basis was adapted to the data using the learning rule given in equation 4.4. The bases were adapted with 50 iterations of simple gradient descent with a batch size of 500 and a step size of 0.1 for the first 30 iterations, which was reduced to 0.001 over the last 20. The most probable coefficients were obtained using a publicly available interior point

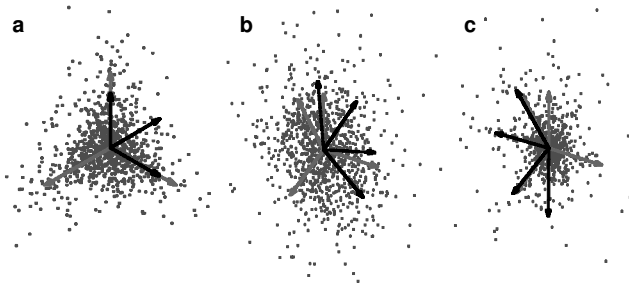


Figure 3: Three examples illustrating the fitting of 2D distributions with over-complete bases. The gray vectors show the true basis vectors used to generate the distribution. The black vectors show the learned basis vectors. For clarity, the vectors have been rescaled. The learned basis vectors can be antiparallel to the true vectors, because both positive and negative coefficients are allowed.

linear programming package (Meszaros, 1997). Convergence of the learning algorithm was rapid, usually reaching the solution in fewer than 30 iterations. A solution was discarded if the magnitude of one of the basis vector dropped to zero. This happened occasionally when one vector was “trapped” between two others that were already representing the data in that region.

The first example is shown in Figure 3a. The data were generated from the true basis vectors (shown in gray) using $\mathbf{x} = \mathbf{A}\mathbf{s}$. To illustrate better the arms of the distribution, the elements of \mathbf{s} were drawn from an exponential distribution (i.e., only positive coefficients) with unit mean. The direction of the learned vectors always matched that of the true generating vectors. The magnitude was smaller than the true basis vectors, possibly due to the approximation used for $P(\mathbf{x}|\mathbf{A})$ (see equation 4.3). Identical results were obtained when the coefficients were generated from a Laplacian prior (i.e., both positive and negative coefficients).

The second example (see Figure 3b) has four arms, again generated from the true basis vectors using a Laplacian (both positive and negative coefficients) with unit variance. The learned vectors were close to the true vectors but showed a consistent bias. This might occur because the data are too dense for there to be any distinct direction or from the approximation to $P(\mathbf{x}|\mathbf{A})$. The bias can be greatly reduced if the coefficients are drawn from a distribution with greater sparsity. The example in Figure 3c uses a data set from the same underlying vectors, but generated from a generalized Laplacian distribution, $P(s_m) \propto \exp(-|s_m|^p)$. When this distribution is fitted to wavelet subband coefficients of images, Buccigrossi and Simoncelli (1997) found that p was in the range [0.5, 1.0]. Varying this exponent varies the kurtosis of the distribution, which is one measure of a sparseness. In

Figure 3c, the data were generated with $p = 0.6$. The arms of this data set are more distinct, and the directions match those of the generating vectors.

6 Quantifying the Efficiency of the Representation

One advantage of the probabilistic framework is that it provides a natural means for comparing objectively alternative representations and models. In this section we compare two methods for comparing the coding efficiency of the learned basis functions.

6.1 Estimating Coding Cost Using $P(\mathbf{x}|\mathbf{A})$. The objective function $P(\mathbf{x}|\mathbf{A})$ for the probability of the data under the model is a natural measure for comparing different models. It is helpful to convert this value into a more intuitive form. According to Shannon's coding theorem, the probability of a code word gives the lower bound on the length of that code word, under the assumption that the model is correct. The number of bits required to encode the pattern is given by

$$\text{\#bits} \geq -\log_2 P(\mathbf{x}|\mathbf{A}) - L \log_2(\sigma_x), \quad (6.1)$$

where L is the dimensionality of the input pattern \mathbf{x} , and σ_x specifies the quantization level of the encoding.

6.1.1 A More Accurate Approximation to $P(\mathbf{x}|\mathbf{A})$. The learning algorithm has been derived by making a gaussian approximation around the posterior maximum $\hat{\mathbf{s}}$. In practice this is sufficiently accurate to generate a useful gradient for learning overcomplete bases. One caution, however, is that there is not necessarily a relation between the local curvature and the volume for general priors, as for a gaussian prior. When comparing alternative models or basis functions, we may want a more accurate approximation. One approach that works well is a method based on second differences.

Rather than rely on the analytic Hessian to estimate the volume around $\hat{\mathbf{s}}$, the volume can be estimated directly by calculating the change in the posterior, $P(\mathbf{s})P(\mathbf{x}|\mathbf{s}, \mathbf{A})$ around the maximum $\hat{\mathbf{s}}$ using second differences. Computing the entire Hessian in this manner would require $O(M^2)$ evaluations of the posterior, which itself is an $O(M^2)$ operation. We can reduce the number of posterior evaluations to $O(M)$ if we estimate the curvature only in the direction of the eigenvectors of the Hessian. To obtain an accurate volume estimate, the volume of gaussian approximation should match as closely as possible to the volume around the posterior maximum. This requires a choice of the step size along the eigenvector direction. In principle, this fit could be optimized, for example, by minimizing the cross entropy, but at significant computational expense. A relatively quick and reliable estimate can be obtained by choosing the step length, η_i , which produces a

fixed drop, Δ , in the value of the posterior log-likelihood,

$$\eta_i = \operatorname{argmin}_{\eta_i} [\log P(\hat{\mathbf{s}}|\mathbf{x}, \mathbf{A}) - \log P(\hat{\mathbf{s}} + \eta_i \mathbf{e}_i|\mathbf{x}, \mathbf{A}) - \Delta], \quad (6.2)$$

where η_i is the step length along the direction of eigenvector \mathbf{e}_i . This method of choosing η_i requires a line search, but this can typically be done with three or four function evaluations. In the examples below, we used $\Delta = 3.8$, the optimal value for approximating a Laplacian with a gaussian. This procedure computes a new Hessian, $\hat{\mathbf{H}}$, which is a more accurate estimate of the posterior curvature. The estimated Hessian is then substituted into (equation 4.3) to obtain a more accurate estimate of $\log P(\mathbf{x}|\mathbf{A})$.

Figure 4 shows cross-sections of the true posterior and the gaussian approximation using the direct Hessian and the second-differences method. The direct Hessian method produced poor approximations in directions where the curvature was sharp around the $\hat{\mathbf{s}}$. Estimating the volume with second differences produced a much better estimate along all of the eigenvector directions. There is no guarantee, however, that this is the best estimate, because there could still exist some directions that are poorly approximated. For example, the first plots in Figures 4a and 4b (labeled $s - \text{smpl}$) show the direction between $\hat{\mathbf{s}}$ using a convergence tolerance of 10^{-4} and $\hat{\mathbf{s}}$ using a convergence tolerance of 10^{-6} . This was the direction of the gradient when optimization was terminated and is likely to be in the direction of coefficients whose values are poorly determined. The maximum has not been reached, and the volume in this direction is underestimated. Overall, however, the second-differences method produces a much more accurate estimate of $\log P(\mathbf{x}|\mathbf{A})$ than that using the analytic Hessian and obtains more accurate estimates of the relative coding efficiencies of different models.

6.1.2 Application to Test Data. We now quantify the efficiency of different representations for various test data sets using the methods described above. The estimated coding cost was calculated from 10 sets of 1000 randomly sampled data points using an encoding precision $\sigma_x = 0.01$. The relative encoding efficiency of the different representations depends on the relative difference of their predictions—how closely $P(\mathbf{x}|\mathbf{A})$ matches that of the underlying distribution. If there is little difference between the predictive distribution, then there will be little difference in the expected coding cost.

Table 1 shows the estimated coding costs of various models for the data sets used above. The uniform distribution gives an upper bound on the coding cost by assuming equal probability for all points within the range spanned by the data points. A simple bivariate gaussian model of the data points, equivalent to a code based on the principal components of the data, does significantly better.

The remaining table entries assume the model defined in equation 2.1 with a Laplacian prior on the basis coefficients. The 2×2 basis matrix is

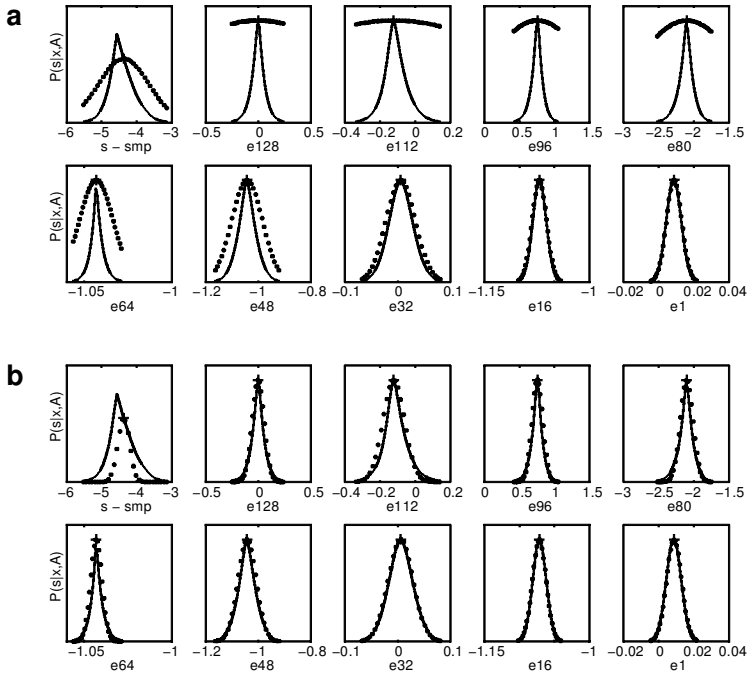


Figure 4: The solid lines in the figure show the cross-sections of a 128-dimensional posterior distribution ($P(s|x, \mathbf{A})$) along the directions of the eigenvectors of the Hessian matrix evaluated at \hat{s} (see equation 2.3). This posterior resulted from fitting a $2 \times$ -overcomplete representation to a 64-sample segment of natural speech (see section 7). The first cross-section is in the direction between \hat{s} using a tolerance of 10^{-4} and a more probable \hat{s} found using a tolerance of 10^{-6} (labeled $s - smp$). The remaining cross-sections are in order of the eigenvalues, showing a sample from smallest ($e128$) to largest ($e1$). The dots show the gaussian approximation for the same cross-section. The + indicates the position of \hat{s} . Note that the x -axes have different scales. (a) The gaussian approximation obtained using the analytic Hessian to estimate the curvature. (b) The same cross-sections but with the gaussian approximation calculated using the second-difference method described in the text.

equivalent to the ICA solution (under the assumption of a Laplacian prior). The 2×3 and 2×4 matrices are overcomplete representations. All bases were learned with the methods discussed above. The most probable coefficients were calculated using the linear programming method as was done during learning.

Table 1 shows that the coding efficiency estimates obtained using the approximation to $P(x|\mathbf{A})$ are reasonably accurate compared to the estimated

Table 1: Estimated Coding Costs using $P(x|\mathbf{A})$.

Model	Bits per Pattern			
	Figure 1a	Figure 3a	Figure 3b	Figure 3c
uniform	22.31 ± 0.17	21.65 ± 0.19	21.57 ± 0.16	26.67 ± 0.31
Gaussian	18.96 ± 0.06	17.98 ± 0.07	18.55 ± 0.06	22.12 ± 0.15
$\mathbf{A}_{2 \times 2}$ (ICA)	18.84 ± 0.04	17.79 ± 0.07	18.38 ± 0.05	21.46 ± 0.08
$\mathbf{A}_{2 \times 3}$	18.59 ± 0.03	16.87 ± 0.06	18.04 ± 0.04	21.12 ± 0.08
$\mathbf{A}_{2 \times 4}$	—	—	18.15 ± 0.08	21.24 ± 0.08

bits per pattern for the uniform and gaussian models for which the expected coding costs are exact. Apart from the uniform model, the differences in the predicted densities are rather subtle and do not show any large differences in coding costs. This would be expected because the densities predicted by the various models are similar. For all the data sets, using a three-vector basis ($\mathbf{A}_{2 \times 3}$) gives greater expected coding efficiencies, which indicates that higher degrees of overcompleteness can yield more efficient coding. For the four-vector basis ($\mathbf{A}_{2 \times 4}$), there is no further improvement in coding efficiency. This reflects the fact that when the prior distribution is limited to a Laplacian, there is an inherent limitation on the model's degrees of freedom; increasing the number of basis functions does not always increase the space of distributions that can be modeled.

6.2 Estimating Coding Cost Using Coefficient Entropy. The probability of the data gives a lower bound on code length but does not specify a code. An alternative method for estimating the coding efficiency is to estimate the entropy of a proposed code. For these models, a natural choice for the code is the vector coefficients, \mathbf{s} . In this approach, the distribution of the coefficients \mathbf{s} (fit to a training data set) is estimated with a function $f(\mathbf{s})$. The coding cost for a test data set is computed estimating the entropy of the fitted coefficients to a quantization level needed to maintain an encoding noise level of σ_x . This has the advantage that $f(\mathbf{s})$ can give a better approximation to the observed distribution of \mathbf{s} (better than the Laplacian assumed under the model). In the examples below, we assume that all the coefficients have the same distribution, although it is straightforward to use a separate function for the distribution estimate of each coefficient. One note of caution with this technique, however, is that it does not include any cost of misfitting. A basis that does not fully span the data space will result in poor fits to the data but can still yield low entropy. If the bases under consideration fully span the data space, then the entropy will yield a reasonable estimate of coding cost.

To compute the entropy, the precision to which each coefficient is encoded needs to be specified. Ideally, $f(\mathbf{s})$ should be quantized to maintain

an encoding noise level of σ_x . In the examples below, we use the approximation $\delta s_i \approx \sigma_x / |\mathbf{A}_i|$. This relates a change in s_i to a change in the data x_i and is exact if \mathbf{A} is orthogonal. We use the mean value of δs_i to obtain a single quantization level δs for all coefficients. The function $f(s)$ by applying kernel density estimation (Silverman, 1986) to the distribution of coefficients fit to a training data set. We use a Laplacian kernel with a window width of $2\delta s$.

The most straightforward method of estimating the coding cost (in bits per pattern) is to sum over the individual entropies,

$$\#bits \geq - \sum_i \frac{n_i}{N} \log_2 f[i], \quad (6.3)$$

where $f(s)$ is quantized as $f[i]$, the index i ranges over all the bins in the quantization, and n_i is the number of counts observed in each bin for each of the coefficients fit to a test data set consisting of N patterns. The distinction between the test and training data set is necessary because the probabilities of the code words need to be specified a priori. Using the same data set for test and training underestimates the cost, although for large data sets, the difference is minimal.

An alternative method of computing the coding cost using the entropy is to make use of the fact that for sparse-overcomplete representations, a subset of the coefficients will be zero. The coding cost can be computed by summing the entropy of the nonzero coefficients plus the cost of identifying them,

$$\#bits \geq - \sum_i \frac{n_i}{N} \log_2 f_{nz}[i] + \min(M - L, L) \log_2(M), \quad (6.4)$$

where $f_{nz}[i]$ is the quantized density estimate of the nonzero coefficients in the i th bin and the index i ranges over all the bins. This method is appropriate when the cost of identifying the nonzero coefficients is small compared to the cost of sending $M - L$ (zero-valued) coefficients.

6.2.1 Application to Test Data. We now quantify the efficiency of various representations for the data sets shown in Figure 3 by computing the entropy of the coefficients. As before, the estimated coding cost was calculated on 1000 randomly sampled data points, using an encoding precision of 0.01. A separate training data set consisting of 10,000 data points was used to estimate the distribution of the coefficients. The entropy was estimated by computing the entropy of the nonzero coefficients (see equation 6.4), which yielded lower estimated coding costs for the overcomplete bases compared to summing the individual entropies (see equation 6.3).

Table 2 shows the estimated coding costs for the same learned bases and data sets used in Table 1. The second column lists the cost of labeling the

Table 2: Estimated Entropy of Nonzero Coefficients.

Model	Labeling Cost	Bits per Pattern			
		Figure 1a	Figure 3a	Figure 3b	Figure 3c
$A_{2 \times 2}$	0.0	18.88 ± 0.04	17.78 ± 0.07	18.18 ± 0.05	21.45 ± 0.09
$A_{2 \times 3}$	1.6	18.02 ± 0.03	17.28 ± 0.05	17.75 ± 0.05	20.95 ± 0.08
$A_{2 \times 4}$	2.0	—	—	17.88 ± 0.07	20.81 ± 0.09

nonzero coefficients. The other columns list the coding cost for the values of the nonzero coefficients. The total coding cost is the sum of these two numbers. For the 2×2 matrices (the ICA solution), the entropy estimate yields approximately the same coding cost as the estimate based on $P(\mathbf{x}|\mathbf{A})$, which indicates that in the complete case, the computation of $P(\mathbf{x}|\mathbf{A})$ is accurate.

This entropy estimate, however, yields a higher coding for the overcomplete bases. The coding cost computed from $P(\mathbf{x}|\mathbf{A})$ suggests that lower average coding costs are achievable. One strategy for lowering the average coding cost would be to avoid identifying the nonzero coefficients for every pattern, for example, by grouping patterns that share common nonzero coefficients. This illustrates the advantages of contrasting the minimal coding cost estimated from the probability with that of a particular code.

7 Learning Sparse Representations of Speech

Speech data were obtained from the TIMIT database, using speech a single speaker, speaking 10 different example sentences. Speech segments were 64 samples in duration (8 msec at the sampling frequency of 8000 Hz). No preprocessing was done. Both a complete ($1 \times$ -overcomplete or 64 basis vectors) and a $2 \times$ -overcomplete basis (128 basis vectors) were learned. The bases were initialized to the standard Fourier basis and a $2 \times$ -overcomplete Fourier basis, which was composed of twice the number of sine and cosine basis functions, linearly spaced in frequency over the Nyquist range.

For larger problems, it is desirable to make the learning faster. Some simple modifications to the basic gradient descent procedure were used that produced more rapid and reliable convergence. For each gradient (see equation A.33), a step size was computed by $\delta_i = \epsilon_i / a_{\max}$, where a_{\max} is the element of the basis matrix, \mathbf{A} , with largest absolute value. The parameter ϵ was reduced from $0.02r$ to $0.001r$ over the first 1000 iterations and fixed at $0.001r$ for the remaining iterations, where r is a measure of the data range and was defined to be the standard deviation of the data.

For the learning rule, we replaced the term \mathbf{A} in equation A.39 with an approximation to $\lambda \mathbf{A} \mathbf{A}^T \mathbf{A} \mathbf{H}^{-1}$ (see equation A.36) as suggested in Lewicki

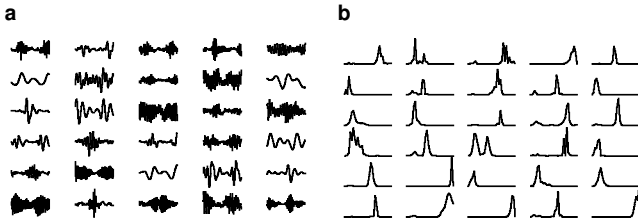


Figure 5: An example of fitting a $2\times$ -overcomplete representation to segments of natural speech. Speech segments consisted of 64 samples in duration (8 msec) at the sampling frequency of 8000 Hz. (a) A random sample of 30 of the 128 basis vectors (each scaled to full range). (b) The power spectral densities (0 to 4000 Hz) of the corresponding basis in a.

and Olshausen (1998, 1999):

$$-\lambda \mathbf{A}^T \mathbf{A} \mathbf{H}^{-1} \approx \mathbf{I} - \mathbf{B} \mathbf{Q} \text{diag}^{-1}[\mathbf{V} + \mathbf{Q}^T \mathbf{B} \mathbf{Q}] \mathbf{Q}^T, \quad (7.1)$$

where $\mathbf{B} = \nabla_{\hat{\mathbf{s}}} \nabla_{\hat{\mathbf{s}}} \log P(\hat{\mathbf{s}})$ and \mathbf{Q} and \mathbf{V} are obtained from the singular value decomposition $\lambda \mathbf{A}^T \mathbf{A} = \mathbf{Q} \mathbf{V} \mathbf{Q}^T$. This yielded solutions similar to equation 4.4, but resulted in fewer zero-length basis vectors. One hundred patterns were used to estimate each learning step. Learning was terminated after 5000 steps.

The most probable basis function coefficients, $\hat{\mathbf{s}}$, were obtained using a modified conjugate gradient routine (Press, Teukolsky, Vetterling, & Flannery, 1992). The basic routine was modified to replace the line search with an approximate Newton step. This approach resulted in a substantial speed improvement and produced much better solutions in a fixed amount of time than the standard routine. To improve speed, a convergence criterion in the conjugate gradient routine was started at value of 10^{-2} and reduced to 10^{-3} over the course of learning.

Figure 5 shows a sample of the learned basis vectors from the $2\times$ -overcomplete basis. The waveforms in Figure 5a show a random sample of the learned basis vectors; the corresponding power spectral densities (0–4000 Hz) are shown in Figure 5b. An immediate observation is that, unlike the Fourier basis vectors, many of the learned basis vectors have a broad bandwidth, and some have multiple spectral peaks. This is an indication that the basis functions contain some of the broadband and harmonic structure inherent in the speech signal.

Another way of contrasting the learned representation with the Fourier representation is shown in Figure 6. This shows the log-coefficient magnitudes over the duration of a speech sentence taken from the TIMIT database for the $2\times$ -overcomplete Fourier (middle plot) and $2\times$ -overcomplete learned

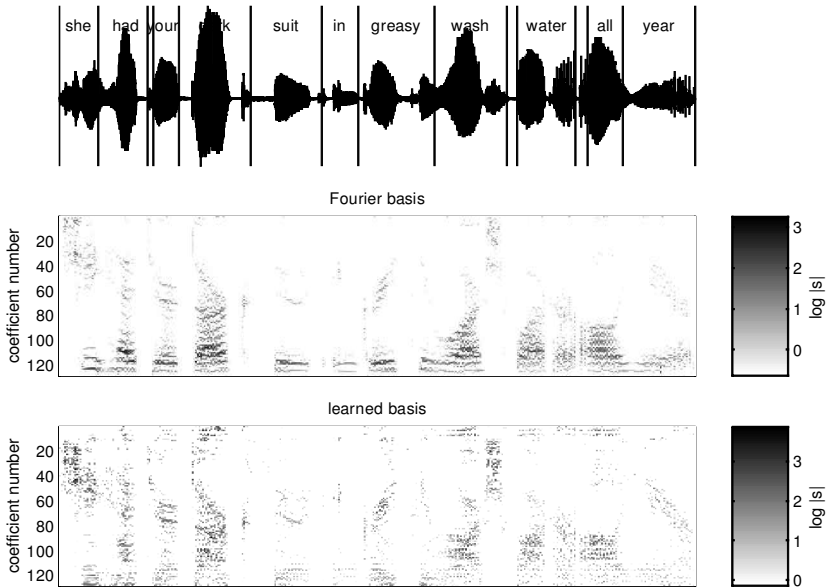


Figure 6: Comparison of coefficient values over the duration of a speech sample from the TIMIT database. (Top) The amplitude envelope of a sentence of speech. The vertical lines are a rough indication of the word boundaries, as listed in the database. (Middle) Plots the log-coefficient magnitudes for a $2\times$ -overcomplete Fourier basis over the duration of the speech example, using nonoverlapping blocks of 64 samples without windowing. (Bottom) Shows the log-coefficient magnitudes for a $2\times$ -overcomplete learned basis. Only the largest 10% of the coefficients (in terms of magnitude) are plotted. The basis vectors are ordered in terms of their power spectra, with higher frequencies toward the top.

representations (bottom plot). The middle plot is similar to a standard spectrogram except that no windowing was performed and the windows were not overlapping.

Figure 6 shows that the coefficient values for the learned basis are used much more uniformly than in the Fourier basis. This is consistent with the learning objective, which optimizes the basis vectors to make the coefficient values as independent as possible. One consequence is that the correlations across frequency that exist in the Fourier representation (reflecting the harmonic structure in the speech) are less apparent in the learned representation.

7.1 Comparing Efficiencies of Representations. Table 3 shows the estimated number of bits per sample to encode a speech segment to a precision of 1 bit out of 8 ($\sigma_x = 1/256$ of the amplitude range). The table shows the

Table 3: Estimated Bits per Sample for Speech Data.

Basis	Estimation Method		
	$-\log_2 P(\mathbf{x} bA) - L \log_2(\sigma_x)$	Total Entropy	Nonzero Entropy
1× learned	4.17 ± 0.07	3.40 ± 0.05	3.36 ± 0.08
1× Fourier	5.39 ± 0.03	4.29 ± 0.07	4.24 ± 0.09
2× learned	5.85 ± 0.06	5.22 ± 0.04	3.07 ± 0.07
2× Fourier	6.87 ± 0.06	6.57 ± 0.14	4.12 ± 0.11

estimated coding costs using the probability of the data (see equation 6.1) and using the entropy, computed by summing the individual entropy estimates of all coefficients (see equation 6.3) (total). Also shown is the entropy of only the nonzero coefficients. By both estimates, the complete and the 2×-overcomplete learned basis perform significantly better than the corresponding Fourier basis. This is not surprising given the observed redundancy in the Fourier representation (see Figure 6).

For all of the bases, the coding cost estimates derived from the entropy of the coefficients are lower than those derived from $P(\mathbf{x}|A)$. A possible reason is that the coefficients are sparser than the Laplacian prior assumed by the model. This is supported by looking at the histogram of the coefficients (see Figure 7), which shows a density with kurtosis much higher than the as-

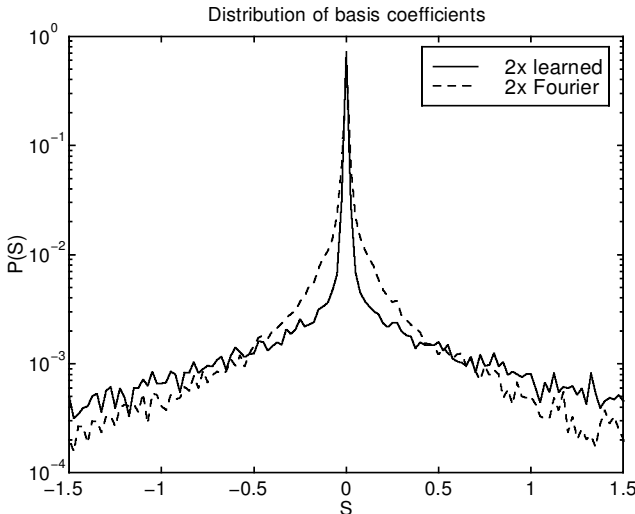


Figure 7: Distribution of basis function coefficients for the 2× learned and Fourier bases. Note the log scale on the y-axis.

sumed Laplacian. The entropy method can obtain a lower estimate, because it fits the observed density of \mathbf{s} .

The last column in the table shows that although the entropy per coefficient of the $2\times$ -overcomplete bases is less than for the complete bases, neither basis yields an overall improvement in coding efficiency. There could be several reasons for this. One likely reason is due to the independence assumption of the model ($P(\mathbf{s}) = \prod_m P(s_m)$). This assumes that there is no structure among the learned basis vectors. From Figure 6, however, it is apparent that there is considerably more structure left in the values of basis vector coefficients. Possible generalizations for capturing this kind of structure are discussed below.

8 Discussion

We have presented an algorithm for learning overcomplete bases. In some cases, overcomplete representations allow a basis to approximate better the underlying statistical density of the data, which can lead to representations that better capture the underlying structure in the data and have greater coding efficiency. The probabilistic formulation of the basis inference problem offers the advantages that assumptions about the prior distribution on the basis coefficients are made explicit. Moreover, by estimating the probability of the data given the model or the entropy of the basis function coefficients, different models can be compared objectively.

This algorithm generalizes ICA so that the model accounts for additive noise and allows the basis to be overcomplete. Unlike standard ICA, where the internal states are computed by inverting the basis function matrix, in this model the transformation from the data to the internal representation is nonlinear. This occurs because when the model is generalized to account for additive noise or when the basis is allowed to be overcomplete, the internal representation is ambiguous. The internal representation is then obtained by maximizing the posterior probability of \mathbf{s} given the assumptions of the model, which is, in general, a nonlinear operation.

A further advantage of this formulation is that it is possible to “denoise” the data. The inference procedure of finding the most probable coefficients automatically separates the data into the underlying signal and the additive noise. By setting the noise to appropriate levels, the model specifies what structure in the data should be ignored and what structure should be modeled by the basis functions. The derivation given here also allows the noise level to be set to zero. In this case, the model attempts to account for all of the variability in the data. This approach to denoising has been applied successfully to natural images (Lewicki & Olshausen, 1998, 1999).

Another potential application is the blind separation of more sources than mixtures. For example, the two-dimensional examples in Figure 3 can be viewed as a source separation problem in which a number of sources are mixed onto a smaller number of channels (three to two in Figure 3a and

four to two in Figures 3b and 3c). The overcomplete bases allow the model to capture the underlying statistical structure in the data space. True source separation will be limited, however, because the sources are being mapped down to a smaller subspace and there is necessarily a loss of information. Nonetheless, is it possible to separate three speakers on two channels with good fidelity (Lee, Lewicki, Girolami, & Sejnowski, 1999).

We have also shown, in the case of natural speech, that learned bases have better coding properties than commonly used representations such as the Fourier basis. In these examples, the learned basis resulted in an estimated coding efficiency for (near-lossless compression) that was about 1.4 bits per sample better than the Fourier basis. This reflects the fact that spectral representations of speech are redundant, because of spectral structures such as speaker harmonics. In the complete case, the model is equivalent to ICA, but with additive noise. We emphasize that these numbers reflect only a very simple coding scheme based on 64 sample speech segments. Other coding schemes can achieve much greater compression. The analysis presented here serves only to compare the relative efficiency of a code based on Fourier representation with that of a learned representation.

The learned overcomplete representations also showed greater coding efficiency than the overcomplete Fourier representation but did not show greater coding efficiency than the complete representation. One possible reason is that the approximations are inaccurate. The approximation used to estimate the probability of the data (see equation 4.2) works well for learning the basis functions and also gives reasonable estimates when exact quantities are available. However, it is possible that the approximation becomes increasingly inaccurate for higher degrees of overcompleteness. An open challenge for this and other inference problems is how to obtain accurate and computationally tractable approximations to the data probability.

Another, perhaps more plausible, reason for the lack of coding efficiency for the overcomplete representations is that the assumptions of the model are inaccurate. An obvious one is the prior distribution for the coefficients. For example, the observed coefficient distribution for the speech data (see Figure 7) is much sparser than that of the Laplacian assumed by the model. Generalizing these models so that $P(\mathbf{s})$ better captures the kind of structure would improve the accuracy of the model and allow it to fit a broader range of distributions. Generalizing these techniques to handle more general prior distributions, however, is not straightforward because of the difficulties in optimization techniques for finding the most probable coefficients and the approximation of the data probability, $\log P(\mathbf{x}|\mathbf{A})$. These are directions we are currently investigating.

We need to question the fundamental assumption that the coefficients are statistically independent. This is not true for structures such as speech, where there is a high degree of nonstationary statistical structure. This type of structure is clearly visible in the spectrogram-like plots in Figure 6. Be-

cause the coefficients are assumed to be independent, it is not possible to capture mutually exclusive relationships that might exist among different speech sounds. For example, different bases could be specialized for different types of phonemes. Capturing this type of structure requires generalizing the model so that it can represent and learn higher-order structure. In applications of overcomplete representations, it is common to assume the coefficients are independent and have Laplacian distributions (Mallat & Zhang, 1993; Chen et al., 1996). These results show that although overcomplete representations can potentially yield many benefits, in practice these can be limited by inaccurate prior assumptions. An improved model of the data distribution will lead to greater coding efficiency and also allow for more accurate inference. The Bayesian framework presented in this article provides a direct method for evaluating the validity of these assumptions and also suggests ways in which these models might be generalized.

Finally, the approach taken here to overcomplete representation may also be useful for understanding the nature of neural codes in the cerebral cortex. One million ganglion cells in the optic nerve are represented by 100 million cells in the primary visual cortex of the monkey. Thus, it is possible that at the first stage of cortical processing, the visual world is overrepresented by a large factor.

Appendix

A.1 Approximating $P(\mathbf{x}|\mathbf{A})$. The integral

$$P(\mathbf{x}|\mathbf{A}) = \int d\mathbf{s} P(\mathbf{s})P(\mathbf{x}|\mathbf{s}, \mathbf{A}) \quad (\text{A.1})$$

can be approximated with the gaussian integral,

$$\int d\mathbf{s} f(\mathbf{s}) \approx f(\hat{\mathbf{s}})(2\pi)^{k/2} |\mathbf{H}(\hat{\mathbf{s}})|^{-1/2}, \quad (\text{A.2})$$

where $|\cdot|$ indicates the determinant and $\nabla_s \nabla_s$ indicates the Hessian, which is evaluated at the solution $\hat{\mathbf{s}}$ (see equation 2.3). Using $f(\mathbf{s}) = P(\mathbf{s})P(\mathbf{x}|\mathbf{s}, \mathbf{A})$ we have

$$\begin{aligned} -\nabla_s \nabla_s [\log P(\mathbf{s})P(\mathbf{x}|\mathbf{s}, \mathbf{A})] &= \mathbf{H}(\mathbf{s}) \\ &= -\nabla_s \nabla_s \left[-\frac{\lambda}{2} (\mathbf{x} - \mathbf{A}\mathbf{s})^2 + \log P(\mathbf{s}) \right] \end{aligned} \quad (\text{A.3})$$

$$= \nabla_s \left[-\lambda \mathbf{A}^T (\mathbf{x} - \mathbf{A}\mathbf{s}) - \nabla_s P(\mathbf{s}) \right] \quad (\text{A.4})$$

$$= \lambda \mathbf{A}^T \mathbf{A} - \nabla_s \nabla_s P(\mathbf{s}). \quad (\text{A.5})$$

A.2 The Hessian of the Laplacian. In the case of a Laplacian prior on s_m ,

$$\log P(\mathbf{s}) \equiv \sum_m \log P(s_m) = M \log \theta - \theta \sum_{m=1}^M |s_m|. \quad (\text{A.6})$$

Because $\log P(\mathbf{s})$ is piece-wise linear in \mathbf{s} , the curvature is zero, and there is a discontinuity in the derivative at zero. To approximate the volume contribution from $P(\mathbf{s})$, we use

$$\frac{\partial}{\partial s_m} \log P(\mathbf{s}) \approx -\theta \tanh(\beta s_m), \quad (\text{A.7})$$

which is equivalent to using the approximation $P(s_m) \propto \cosh^{-\theta/\beta}(\beta s_m)$. For large β this approximates the true Laplacian prior while staying smooth around zero. This leads to the following diagonal expression for the Hessian:

$$\frac{\partial^2}{\partial s_m^2} \log P(s_m) = -\theta \beta \operatorname{sech}^2(\beta s_m). \quad (\text{A.8})$$

A.3 Derivation of the Learning Rule. For a set of independent data vectors $\mathbf{X} = \mathbf{x}_1, \dots, \mathbf{x}_K$, expand the posterior probability density by a saddle-point approximation:

$$\begin{aligned} \log P(\mathbf{X}|\mathbf{A}) &= \log \prod_k P(\mathbf{x}_k|\mathbf{A}) \\ &\approx \frac{KL}{2} \log \frac{\lambda}{2\pi} + \frac{KM}{2} \log(2\pi) \\ &\quad + \sum_{k=1}^K \left[\log P(\hat{\mathbf{s}}_k) - \frac{\lambda}{2} (\mathbf{x}_k - \mathbf{A}\hat{\mathbf{s}}_k)^2 - \frac{1}{2} \log \det \mathbf{H}(\hat{\mathbf{s}}_k) \right]. \end{aligned} \quad (\text{A.9})$$

A learning rule can be obtained by differentiating $\log P(\mathbf{x}|\mathbf{A})$ with respect to \mathbf{A} . Letting $\nabla_A = \partial/\partial \mathbf{A}$ and for clarity letting $\mathbf{H} = \mathbf{H}(\mathbf{s})$, we see that there are three main derivatives that need to be considered:

$$\nabla_A \log P(\mathbf{x}|\mathbf{A}) = \nabla_A \log P(\hat{\mathbf{s}}) - \frac{\lambda}{2} \nabla_A (\mathbf{x} - \mathbf{A}\hat{\mathbf{s}})^2 - \frac{1}{2} \nabla_A \log \det \mathbf{H}. \quad (\text{A.10})$$

We consider each in turn.

A.3.1 Deriving $\nabla_A \log P(\hat{\mathbf{s}})$. The first term in the learning rule specifies how to change \mathbf{A} to make the representation $\hat{\mathbf{s}}$ more probable. If we assume a prior distribution with high kurtosis, this component will change the weights to make the representation sparser.

From the chain rule,

$$\frac{\partial \log P(\hat{\mathbf{s}})}{\partial \mathbf{A}} = \sum_m \frac{\partial \log P(\hat{\mathbf{s}}_m)}{\partial \hat{\mathbf{s}}_m} \frac{\partial \hat{\mathbf{s}}_m}{\partial \mathbf{A}}, \quad (\text{A.11})$$

assuming $P(\hat{\mathbf{s}}) = \prod_m P(\hat{\mathbf{s}}_m)$. To obtain $\partial \hat{\mathbf{s}}_m / \partial \mathbf{A}$, we need to describe $\hat{\mathbf{s}}$ as a function of \mathbf{A} . If the basis is complete (and we assume low noise), then we can simply invert \mathbf{A} to obtain $\hat{\mathbf{s}} = \mathbf{A}^{-1}\mathbf{x}$. When \mathbf{A} is overcomplete, however, there is no simple expression, but we can still make an approximation.

For certain priors, the most probable solution, $\hat{\mathbf{s}}$, will yield at most L nonzero elements. In effect, the procedure for computing $\hat{\mathbf{s}}$ selects a complete basis from \mathbf{A} that best accounts for the data \mathbf{x} . We can use this hypothetical reduced basis to derive a learning algorithm for the full basis matrix \mathbf{A} . Let $\check{\mathbf{A}}$ represent a reduced basis for a particular \mathbf{x} , that is, $\check{\mathbf{A}}$ is composed of the basis vectors in \mathbf{A} that have nonzero coefficients. More precisely, let c_1, \dots, c_L be the indices of the nonzero coefficients in $\hat{\mathbf{s}}$. If there are fewer than L nonzero coefficients, zero-valued coefficients can be included without loss of generality. Then, $\check{\mathbf{A}}_{1:L,i} \equiv \mathbf{A}_{1:L,c_i}$. We then have

$$\check{\mathbf{s}} = \check{\mathbf{A}}^{-1}(\mathbf{x} - \boldsymbol{\epsilon}), \quad (\text{A.12})$$

where $\check{\mathbf{s}}$ is equal to $\hat{\mathbf{s}}$ with $M-L$ zero-valued elements removed. $\check{\mathbf{A}}^{-1}$ obtained by removing the columns of \mathbf{A} corresponding to the $M-L$ zero-valued elements of $\hat{\mathbf{s}}$. This allows the use of results obtained for the case when \mathbf{A} is invertible. Note that the construction of the reduced basis is only a mathematical device used for this derivation. The final gradient equation does not depend on this construction. Following MacKay (1996) we have

$$\frac{\partial \check{s}_k}{\partial \check{a}_{ij}} = \frac{\partial}{\partial \check{a}_{ij}} \sum_l \check{\mathbf{A}}_{kl}^{-1} (x_l - \epsilon_l), \quad (\text{A.13})$$

Using the identity $\partial \mathbf{A}_{kl}^{-1} / \partial a_{ij} = -\mathbf{A}_{ki}^{-1} \mathbf{A}_{jl}^{-1}$,

$$\frac{\partial \check{s}_k}{\partial \check{a}_{ij}} = - \sum_l \check{\mathbf{A}}_{ki}^{-1} \check{\mathbf{A}}_{jl}^{-1} (x_l - \epsilon_l). \quad (\text{A.14})$$

$$= -\check{\mathbf{A}}_{ki}^{-1} \check{s}_j. \quad (\text{A.15})$$

Letting $\check{z}_k = \partial \log P(\check{s}_k) / \partial \check{s}_k$,

$$\frac{\partial \log P(\check{\mathbf{s}})}{\partial \check{a}_{ij}} = - \sum_k \check{z}_k \check{\mathbf{A}}_{ki}^{-1} \check{s}_j. \quad (\text{A.16})$$

Changing back to matrix notation,

$$\frac{\partial \log P(\check{\mathbf{s}})}{\partial \check{\mathbf{A}}} = -\check{\mathbf{A}}^{-T} \check{\mathbf{z}} \check{\mathbf{s}}^T. \quad (\text{A.17})$$

This derivative can be expressed in terms of the original variables (that is, nonreduced). We invert the mapping used to obtain the reduced coefficients, $\hat{\mathbf{s}} \rightarrow \check{\mathbf{s}}$. We have $z_{c_m} = \check{z}_m$, with the remaining $(M - L)$ values of \mathbf{z} defined to be zero. We define the matrix $\mathbf{W}_{1:L,c_i} \equiv \check{\mathbf{A}}_{1:L,i}^{-1}$, with the remaining rows equal to zero. Then

$$\frac{\partial \log P(\mathbf{s})}{\partial \mathbf{A}} = -\mathbf{W}^T \mathbf{z} \mathbf{s}^T. \quad (\text{A.18})$$

This gives a gradient of zero for the columns of \mathbf{A} that have zero-valued coefficients.

A.3.2 Deriving $\nabla_A(\mathbf{x} - \mathbf{A}\hat{\mathbf{s}})^2$. The second term specifies how to change \mathbf{A} to minimize the data misfit. Letting $e_k = [\mathbf{x} - \mathbf{A}\hat{\mathbf{s}}]_k$ and using the results and notation from above:

$$\frac{\partial}{\partial a_{ij}} \frac{\lambda}{2} \sum_k e_k^2 = \lambda e_i s_j + \lambda \sum_k e_k \sum_l a_{kl} \frac{\partial s_l}{\partial a_{ij}} \quad (\text{A.19})$$

$$= \lambda e_i s_j + \lambda \sum_k e_k \sum_l -a_{kl} w_{li} s_j \quad (\text{A.20})$$

$$= \lambda e_i s_j - \lambda e_i s_j = 0. \quad (\text{A.21})$$

Thus, there is no gradient component arising from the error term that might be expected if the residual error has no structure.

A.3.3 Deriving $\nabla_A \log \det \mathbf{H}$. The third term in the learning rule specifies how to change the weights to minimize the width of the posterior distribution $P(\mathbf{x}|\mathbf{A})$ and thus increase the overall probability of the data.

Using the chain rule,

$$\frac{\partial \log \det \mathbf{H}}{\partial a_{ij}} = \frac{1}{\det \mathbf{H}} \sum_{mn} \frac{\partial \det \mathbf{H}}{\partial \mathbf{H}_{mn}} \frac{\partial \mathbf{H}_{mn}}{\partial a_{ij}}. \quad (\text{A.22})$$

An element of \mathbf{H} is defined by

$$\mathbf{H}_{mn} = \sum_k \lambda a_{km} a_{kn} + b_{mn} \quad (\text{A.23})$$

$$= c_{mn} + b_{mn}, \quad (\text{A.24})$$

where $b_{mn} = (-\nabla_s \nabla_s \log P(\hat{\mathbf{s}}))_{mn}$. Using the identity $\partial \det \mathbf{H} / \partial \mathbf{H}_{mn} = (\det \mathbf{H}) \mathbf{H}_{mn}^{-1}$ we obtain

$$\frac{\partial \log \det \mathbf{H}}{\partial a_{ij}} = \sum_{mn} \mathbf{H}_{mn}^{-1} \left[\frac{\partial c_{mn}}{\partial a_{ij}} + \frac{\partial b_{mn}}{\partial a_{ij}} \right]. \quad (\text{A.25})$$

Considering the first term in the square brackets,

$$\frac{\partial c_{mn}}{\partial a_{ij}} = \begin{cases} 0 & m, n \neq j, \\ \lambda a_{in} & m = j, \\ \lambda a_{im} & n = j, \\ 2\lambda a_{ij} & m = n = j. \end{cases} \quad (\text{A.26})$$

We then have

$$\sum_{mn} \mathbf{H}_{mn}^{-1} \frac{\partial c_{mn}}{\partial a_{ij}} = \sum_{m \neq j} \mathbf{H}_{mj}^{-1} \lambda a_{im} + \sum_{m \neq j} \mathbf{H}_{jm}^{-1} \lambda a_{im} + \mathbf{H}_{jj}^{-1} 2\lambda a_{ij} \quad (\text{A.27})$$

$$= 2\lambda \sum_m \mathbf{H}_{mj}^{-1} a_{im}, \quad (\text{A.28})$$

using the fact that $\mathbf{H}_{mj}^{-1} = \mathbf{H}_{jm}^{-1}$ due to the symmetry of the Hessian. In matrix notation this becomes

$$\sum_{mn} \mathbf{H}_{mn}^{-1} \frac{\partial c_{mn}}{\partial \mathbf{A}} = 2\lambda \mathbf{A} \mathbf{H}^{-1}. \quad (\text{A.29})$$

Next we derive $\partial b_{mn}/\partial a_{ij}$. Assuming $P(\hat{\mathbf{s}}) = \prod_m P(\hat{s}_m)$ we have that $\nabla_s \nabla_s \log P(\hat{\mathbf{s}})$ is diagonal and thus

$$\sum_{mn} \mathbf{H}_{nm}^{-1} \frac{\partial b_{mn}}{\partial a_{ij}} = \sum_m \mathbf{H}_{mm}^{-1} \frac{\partial b_{mm}}{\partial \hat{s}_m} \frac{\partial \hat{s}_m}{\partial a_{ij}}. \quad (\text{A.30})$$

Letting $2y_m = \mathbf{H}_{mm}^{-1} \partial b_{mm}/\partial \hat{s}_m$ and using the result under the reduced representation (see equation A.15),

$$\sum_{mn} \mathbf{H}_{nm}^{-1} \frac{\partial b_{mn}}{\partial \mathbf{A}} = -2\mathbf{W}^T \mathbf{y} \hat{\mathbf{s}}^T. \quad (\text{A.31})$$

Finally, putting the results for $\partial c_{mn}/\partial \mathbf{A}$ and $\partial b_{mn}/\partial \mathbf{A}$ back into equation A.25,

$$\frac{\partial \log \det \mathbf{H}}{\partial \mathbf{A}} = 2\lambda \mathbf{A} \mathbf{H}^{-1} - 2\mathbf{W}^T \mathbf{y} \hat{\mathbf{s}}^T. \quad (\text{A.32})$$

Gathering the three components of $\nabla_A \log P(\mathbf{x}|\mathbf{A})$ together yields the following expression for the learning rule:

$$\nabla_A \log P(\mathbf{x}|\mathbf{A}) = -\mathbf{W}^T \mathbf{z} \hat{\mathbf{s}}^T - \lambda \mathbf{A} \mathbf{H}^{-1} + \mathbf{W}^T \mathbf{y} \hat{\mathbf{s}}^T. \quad (\text{A.33})$$

A.3.4 Stabilizing and Simplifying the Learning Rule. Using the gradient given in equation A.33 directly is problematic due to the matrix inverses, which make it both impractical and unstable. This can be alleviated, however, by multiplying the gradient by an appropriate positive definite matrix (Amari et al., 1996). This rescales the components of the gradient but still preserves a direction valid for optimization. The fact that $\mathbf{A}^T \mathbf{W}^T = \mathbf{I}$ for all \mathbf{W} allows us to eliminate \mathbf{W} from the equation for the learning rule:

$$\mathbf{A} \mathbf{A}^T \nabla_A \log P(\mathbf{x}|\mathbf{A}) = -\mathbf{A} \mathbf{z} \hat{\mathbf{s}}^T - \lambda \mathbf{A} \mathbf{A}^T \mathbf{A} \mathbf{H}^{-1} + \mathbf{A} \mathbf{y} \hat{\mathbf{s}}^T. \quad (\text{A.34})$$

Additional simplifications can be made. If λ is large (low noise), then the Hessian is dominated by $\lambda \mathbf{A}^T \mathbf{A}$ and

$$-\lambda \mathbf{A} \mathbf{A}^T \mathbf{A} \mathbf{H}^{-1} = -\mathbf{A} \lambda \mathbf{A}^T \mathbf{A} (\lambda \mathbf{A}^T \mathbf{A} + \mathbf{B})^{-1} \quad (\text{A.35})$$

$$\approx -\mathbf{A}. \quad (\text{A.36})$$

It is also possible to obtain more accurate approximations of this term (Lewicki & Olshausen, 1998, 1999).

The vector \mathbf{y} hides a computation involving the inverse Hessian. If the basis vectors in \mathbf{A} are randomly distributed, then as the dimensionality of \mathbf{A} increases, the basis vectors become approximately orthogonal, and consequently the Hessian becomes approximately diagonal. In this case,

$$y_m \approx \frac{1}{\mathbf{H}_{mm}} \frac{\partial b_{mm}}{\partial \hat{s}_m} \quad (\text{A.37})$$

$$= \frac{\partial b_{mm} / \partial \hat{s}_m}{\lambda \sum_k a_{km}^2 + b_{mm}}. \quad (\text{A.38})$$

Thus if $\log P(\mathbf{s})$ and its derivatives are smooth, y_m vanishes for large λ . In practice, excluding \mathbf{y} from the learning rule yields more stable learning. We conjecture that this term can be ignored, possibly because it represents curvature components that are unrelated to the volume.

We thus obtain the following expression for a learning rule:

$$\Delta \mathbf{A} = \mathbf{A} \mathbf{A}^T \nabla_A \log P(\mathbf{x}|\mathbf{A}) \approx -\mathbf{A} \mathbf{z} \hat{\mathbf{s}}^T - \mathbf{A} \quad (\text{A.39})$$

$$= -\mathbf{A} (\mathbf{z} \hat{\mathbf{s}}^T + \mathbf{I}). \quad (\text{A.40})$$

Acknowledgments

We thank Bruno Olshausen and Tony Bell for many helpful discussions.

References

- Amari, S., Cichocki, A., & Yang, H. H. (1996). A new learning algorithm for blind signal separation. In D. Touretzky, M. Mozer, & M. Hasselmo (Eds.), *Advances in neural and information processing systems*, 8 (pp. 757–763). San Mateo, CA: Morgan Kaufmann.
- Atick, J. J. (1992). Could information-theory provide an ecological theory of sensory processing? *Network-Computation in Neural Systems*, 3(2), 213–251.
- Barlow, H. B. (1961). Possible principles underlying the transformation of sensory messages. In W. A. Rosenbluth (Ed.), *Sensory communication* (pp. 217–234). Cambridge, MA: MIT Press.
- Barlow, H. B. (1989). Unsupervised learning. *Neural Computation*, 1, 295–311.
- Bell, A. J., & Sejnowski, T. J. (1995). An information maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6), 1129–1159.
- Buccigrossi, R. W., & Simoncelli, E. P. (1997). *Image compression via joint statistical characterization in the wavelet domain* (Tech. Rep. No. 414). Philadelphia: University of Pennsylvania.
- Cardoso, J.-F. (1997). Infomax and maximum likelihood for blind source separation. *IEEE Signal Processing Letters*, 4, 109–111.
- Chen, S., Donoho, D. L., & Saunders, M. A. (1996). *Atomic decomposition by basis pursuit* (Technical Rep.). Stanford, CA: Department of Statistics, Stanford University.
- Coifman, R. R., & Wickerhauser, M. V. (1992). Entropy-based algorithms for best basis selection. *IEEE Transactions on Information Theory*, 38(2), 713–718.
- Comon, P. (1994). Independent component analysis, a new concept. *Signal Processing*, 36(3), 287–314.
- Daubechies, I. (1990). The wavelet transform, time-frequency localization, and signal analysis. *IEEE Transactions on Information Theory*, 36(5), 961–1004.
- Daugman, J. G. (1988). Complete discrete 2-D Gabor transforms by neural networks for image-analysis and compression. *IEEE Transactions on Acoustics Speech and Signal Processing*, 36(7), 1169–1179.
- Daugman, J. G. (1989). Entropy reduction and decorrelation in visual coding by oriented neural receptive-fields. *IEEE Trans. Bio. Eng.*, 36(1), 107–114.
- Field, D. J. (1994). What is the goal of sensory coding? *Neural Computation*, 6(4), 559–601.
- Hinton, G. E., & Sejnowski, T. J. (1986). Learning and relearning in Boltzmann machines. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing* (Vol. 1, pp. 282–317). Cambridge, MA: MIT Press.
- Jutten, C., & Héroult, J. (1991). Blind separation of sources. 1. An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24(1), 1–10.
- Lee, T.-W., Lewicki, M., Girolami, M., & Sejnowski, T. (1999). Blind source separation of more sources than mixtures using overcomplete representations. *IEEE Sig. Proc. Lett.*, 6, 87–90.
- Lewicki, M. S., & Olshausen, B. A. (1998). Inferring sparse, overcomplete image codes using an efficient coding framework. In M. Kearns, M. Jordan, & S. Solla

- (Eds.), *Advances in neural and information processing systems*, 10. San Mateo, CA: Morgan Kaufmann.
- Lewicki, M. S., & Olshausen, B. A. (1999). A probabilistic framework for the adaptation and comparison of image codes. *J. Opt. Soc. Am. A: Optics, Image Science, and Vision*, 16(7), 1587–1601.
- Linsker, R. (1988). Self-organization in a perceptual network. *Computer*, 21(3), 105–117.
- MacKay, D. J. C. (1996). *Maximum likelihood and covariant algorithms for independent component analysis*. Unpublished manuscript. Cambridge: University of Cambridge, Cavendish Laboratory. Available online at: <ftp://wol.ra.phy.cam.ac.uk/pub/mackay/ica.ps.gz>.
- Makeig, S., Jung, T. P., Bell, A. J., Ghahremani, D., & Sejnowski, T. J. (1996). Blind separation of event-related brain response components. *Psychophysiology*, 33, S58–S58.
- Mallat, S. G., & Zhang, Z. F. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12), 3397–3415.
- McKeown, M. J., Jung, T.-P., Makeig, S., Brown, G. G., Kindermann, S. S., Lee, T., & Sejnowski, T. (1998). Spatially independent activity patterns in functional magnetic resonance imaging data during the stroop color-naming task. *Proc. Natl. Acad. Sci. USA*, 95, 803–810.
- Meszaros, C. (1997). BPMPD: An interior point linear programming solver. Code available online at: <ftp://ftp.netlib.org/opt/bpmpd.tar.gz>.
- Nadal, J.-P. & Parga, N. (1994a). Nonlinear neurons in the low-noise limit: A factorial code maximizes information transfer. *Network*, 5, 565–581.
- Nadal, J.-P., & Parga, N. (1994b). Redundancy reduction and independent component analysis: Conditions on cumulants and adaptive approaches. *Network*, 5, 565–581.
- Olshausen, B. A., & Field, D. J. (1996). Emergence of simple-cell receptive-field properties by learning a sparse code for natural images. *Nature*, 381, 607–609.
- Olshausen, B. A., & Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Res.*, 37(23).
- Osuna, E., Freund, R., & Girosi, F. (1997). An improved training algorithm for support vector machines. *Proc. of IEEE NNSP'97* (pp. 24–26).
- Pearlmutter, B. A., & Parra, L. C. (1997). Maximum likelihood blind source separation: A context-sensitive generalization of ICA. In M. Mozer, M. Jordan, & T. Petsche (Eds.), *Advances in neural and information processing systems*, 9. San Mateo, CA: Morgan Kaufmann.
- Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical recipes in C: The art of scientific programming* (2nd ed.). Cambridge: Cambridge University Press.
- Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. New York: Chapman and Hall.
- Simoncelli, E. P., Freeman, W. T., Adelson, E. H., & Heeger, D. J. (1992). Shiftable multiscale transforms. *IEEE Trans. Info. Theory*, 38, 587–607.