Learning Sparse Overcomplete Codes for Images

Joseph F. Murray
Massachusetts Institute of Technology
Brain and Cognitive Sciences Department
77 Massachusetts Ave. 46-5065
Cambridge, MA 02139
and
Kenneth Kreutz-Delgado
University of California, San Diego
Electrical and Computer Engineering Department
9500 Gilman Dr. Dept 0407
La Jolla, Ca 92093-0407
Email: jfmurray@jfmurray.org, kreutz@ece.ucsd.edu

December 13, 2006

Abstract

Images can be coded accurately using a sparse set of vectors from a learned overcomplete dictionary, with potential applications in image compression and feature selection for pattern recognition. We present a survey of algorithms that perform dictionary learning and sparse coding and make three contributions. First, we compare our overcomplete dictionary learning algorithm (FOCUSS-CNDL) with overcomplete independent component analysis (ICA). Second, noting that once a dictionary has been learned in a given domain the problem becomes one of choosing the vectors to form an accurate, sparse representation, we compare a recently developed algorithm (sparse Bayesian learning with adjustable variance Gaussians, SBL-AVG) to well known methods of subset selection: matching pursuit and FOCUSS. Third, noting that in some cases it may be necessary to find a non-negative sparse coding, we present a modified version of the FOCUSS algorithm that can find such non-negative codings. Efficient parallel implementations in VLSI could make these algorithms more practical for many applications.

1 Introduction

Most modern lossy image and video compression standards have as a basic component the transformation of small patches of the image. The discrete cosine transform (DCT) is the most popular, and is used in the JPEG and MPEG compression standards [1]. The DCT uses a fixed set of basis vectors (discrete cosines of varying spatial frequencies) to represent each image patch, which is typically 8x8 pixels. In recent years, many algorithms have been developed that learn a transform basis adapted to the statistics of the input signals. Two widely used basis-learning algorithms are *principal component analysis* (PCA), which finds an orthogonal basis using second-order statistics [2], and *independent component analysis* (ICA) which finds a non-orthogonal representation using higher order statistics [3, 4]. The set of bases used by PCA and ICA are complete or undercomplete, i.e. the matrix defining the transformation

$A \in \mathbb{R}^{m \times n}$ has $m \geq n$, implying that the output has the same or lower dimensionality as the input. Newer classes of algorithms [5, 6] allow the use of an overcomplete A, which we will refer to as a *dictionary* to distinguish it from a basis, which must by definition be linearly independent (although some authors use the term *basis* even when referring to an overcomplete set). Dictionaries are also referred to as *frames* [7].

We discuss the problem of representing images with a highly sparse set of vectors drawn from a learned overcomplete dictionary. The problem has received considerable attention since the work of Olshausen and Field [8], who suggest that this is the strategy used by the visual cortex for representing images. The implication is that a sparse, overcomplete representation is especially suitable for visual tasks such as object detection and recognition that occur in higher regions of the cortex. Non-learned dictionaries (often composed of Gabor functions) are used to generate the features used in many pattern recognition systems [9], and we believe that recognition performance could be improved by using learned dictionaries that are adapted to the image statistics of the inputs.

The sparse overcomplete coding problem has two major parts: learning the dictionary adapted to the input environment, and sparsely coding new patterns using that dictionary. We present and compare experimentally algorithms for both of these tasks. In Section 2, we discuss sparse coding assuming a known, fixed dictionary using the following algorithms: focal-underdetermined system solver (FOCUSS) [10], sparse Bayesian learning with adjustable-variance Gaussians (SBL-AVG) [11] and modified matching pursuit (MMP) [12]. With earlier algorithms such as PCA, ICA and DCT transforms, finding the coefficients requires only a matrix multiply, however with an overcomplete dictionary the representation of a signal is underdetermined, so an additional criteria such as sparseness must be used. In Section 3, we discuss algorithms for learning the dictionary: FOCUSS-CNDL (column-normalized dictionary learning) [13, 5], and an overcomplete extension of ICA [14]. Section 4 explains the evaluation method for comparing image codings, and Section 5 presents the experimental results.

A key result of work in sparse overcomplete coding is that images (and other data) can be coded more efficiently using a learned dictionary than with a non-adapted basis (e.g. DCT, wavelet or Gabor) [14, 15]. For example, it is shown in [15] (see their Table 5.3) that with from 1 to 12 vectors per image patch, the distortion with learned dictionaries is less than with DCT. Our earlier work using learned dictionaries has shown that overcomplete codes can be more efficient than learned complete codes in terms of entropy (bits/pixel), even though there are many more coefficients than image pixels in an overcomplete coding [5]. When sparse overcomplete dictionaries are used in complete compression systems, they have shown improved compression over standard techniques. A compression system based on methods closely related to those presented here was shown to improve performance over JPEG for bit rates of 0.4 bits/pixel and lower [7]. The tradeoff for this increased compression is that overcomplete coding is more computationally demanding, but since the algorithms are based on matrix algebra they are easily parallelizable and have potential for implementation in DSP or custom VLSI hardware, as discussed in Section 6. Sparse coding has many other applications in signal processing including high-resolution spectral estimation, direction-of-arrival estimation, speech coding, biomedical imaging and function approximation (see [10] for more references to these).

In some problems, we may desire (or the physics of the problem may dictate) non-negative sparse codings. An example of such a problem is modeling pollution, where the amount of pollution from any particular factory is non-negative [16]. Methods for non-negative matrix factorization were developed [17] and applied to images and text, with the additional constraint that dictionary elements also be non-negative. A multiplicative algorithm for non-negative coding was developed and applied to images in [18]. A non-negative Independent Component Analysis (ICA) algorithm was presented in [19] (which also discusses other applications). In

[18, 19] only the complete case was considered. Here, in Section 2.1, we present an algorithm that can learn non-negative sources from an overcomplete dictionary, which leads naturally to a learning method that adapts the dictionary for such sources.

2 Sparse Coding and Vector Selection

The problem of sparse coding is that of representing some data $\mathbf{y} \in \mathbb{R}^m$ (e.g. a patch of an image) using a small number of non-zero components in a source vector $\mathbf{x} \in \mathbb{R}^n$ under the linear generative model

$$\mathbf{y} = A\mathbf{x} + \nu\,, \tag{1}$$

where the full-row rank dictionary $A \in \mathbb{R}^{m\times n}$ may be overcomplete ($n > m$), and the additive noise ν is assumed to be Gaussian, $p_\nu = \mathcal{N}(0, \sigma_\nu^2)$. By assuming a prior $p_X(\mathbf{x})$ on the sources, we can formulate the problem in a Bayesian framework and find the maximum *a posteriori* solution for $\mathbf{x}$,

$$\begin{aligned} \widehat{\mathbf{x}} &= \arg\max_{\mathbf{x}} p(\mathbf{x}|A,\mathbf{y}) \\ &= \arg\max_{\mathbf{x}} [\log p(\mathbf{y}|A,\mathbf{x}) + \log p_X(\mathbf{x})]\,. \end{aligned} \tag{2}$$

By making an appropriate choice for the prior $p_X(\mathbf{x})$, we can find solutions with high sparsity (i.e. few non-zero components). We define *sparsity* as the number of elements of $\mathbf{x}$ that are zero, and the related quantity *diversity* as the number of non-zero elements, so that diversity = (n − sparsity). Assuming the prior of the sources $\mathbf{x}$ is a generalized exponential distribution of the form,

$$p_X(\mathbf{x}) = ce^{-\lambda\, d_p(\mathbf{x})}\,, \tag{3}$$

where the parameter λ and function $d_p(\mathbf{x})$ determine the shape of distribution and c is a normalizing constant to ensure $p_X(\mathbf{x})$ is a density function. A common choice for the prior on $\mathbf{x}$ is for the function $d_p(\mathbf{x})$ to be the p-norm-like measure,

$$d_p(\mathbf{x}) = \|\mathbf{x}\|_p^p = \sum_{i=1}^{n} |x_i|^p\,, \quad 0 \le p \le 1\,, \tag{4}$$

where x_i are the elements of the vector $\mathbf{x}$. (Note that for $p < 1$, $\|\mathbf{x}\|_p = (d_p(\mathbf{x}))^{\frac{1}{p}}$ is *not* a norm.) When $p = 0$, $d_p(\mathbf{x})$ is a count of the number of non-zero elements of $\mathbf{x}$ (diversity), and so $d_p(\mathbf{x})$ is referred to as a *diversity measure* [5].

With these choices for $d_p(\mathbf{x})$ and p_ν, we find that,

$$\begin{aligned} \widehat{\mathbf{x}} &= \arg\max_{\mathbf{x}} [\log p(\mathbf{y}|A,\mathbf{x}) + \log p_X(\mathbf{x})] \\ &= \arg\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|^2 + \lambda\|\mathbf{x}\|_p^p\,. \end{aligned} \tag{5}$$

The parameter λ can be seen as a regularizer that adjusts the tradeoff between sparse solutions (high λ) and accurate reconstruction (low λ). In the limit that $p \to 0$ we obtain an optimization problem that directly minimizes the reconstruction error and the diversity of $\mathbf{x}$. When $p = 1$ the problem no longer directly minimizes diversity, but the right-hand-side of (5) has the desirable property of being globally convex and so has no local minima. The $p = 1$ cost function is used in *basis pursuit* [20], where the resulting linear programming problem is usually solved with an interior point method.

Some recent theoretical results have determined conditions under which the $p = 1$ (basis pursuit) solution finds the true ($p = 0$) sparsest solution [21]. However, an evaluation of these

bounds has shown that the conditions are restrictive, and that in fact the global optima associated with $p = 1$ only finds the sparsest solution when that sparsity is very high [22]. These results and related experiments show that in practice the $p = 1$ cost function does not always correspond with the sparsest solution, and that $p < 1$ often provides a more desirable cost function [23].

2.1 FOCUSS and Non-negative FOCUSS

For a given, known dictionary A, the *focal underdetermined system solver* (FOCUSS) was developed to solve (5) for $p \leq 1$ [24, 10]. The FOCUSS algorithm was first applied to the problem of magnetoencephalography (MEG), where spatially localized signals in the brain mix with each other before reaching the sensors, leading to the related problems of localizing the sources and removing undesired artifacts [24, 25].

FOCUSS is an iterative re-weighted factored-gradient approach, and has consistently shown better performance than greedy vector-selection algorithms such as basis pursuit and matching pursuit, although at a cost of increased computation [10]. Previous versions of FOCUSS have assumed that $\mathbf{x}$ is unrestricted on $\mathbb{R}^n$. In some cases however, we may require that the sources be non-negative, $x_i \geq 0$. This amounts to a change of prior on $\mathbf{x}$ from symmetric to one-sided, but this results in nearly the same optimization problem as (5). To create a non-negative FOCUSS algorithm, we need to ensure that the x_i are initialized to non-negative values, and that each iteration keeps the sources in the feasible region. To do so, proposing a one-sided (asymmetrical) diversity measure $d_p(\mathbf{x})$, the *non-negative* FOCUSS algorithm can be derived,

$$\begin{aligned}
\Pi^{-1}(\widehat{\mathbf{x}}) &= \operatorname{diag}(|\widehat{x}_i|^{2-p}) \\
\lambda &= \lambda_{\max}\left(1 - \frac{\|\mathbf{y} - A\widehat{\mathbf{x}}\|}{\|\mathbf{y}\|}\right), \quad \lambda > 0 \\
\widehat{\mathbf{x}} &\leftarrow \Pi^{-1}(\widehat{\mathbf{x}})A^T\left(\lambda I + A\Pi^{-1}(\widehat{\mathbf{x}})A^T\right)^{-1}\mathbf{y} \\
\widehat{x}_i &\leftarrow \begin{cases} 0 & \widehat{x}_i < 0 \\ \widehat{x}_i & \widehat{x}_i \geq 0 \end{cases},
\end{aligned} \tag{6}$$

where λ is a heuristically-adapted regularization term, limited by $\lambda_{\max}$ which controls the tradeoff between sparsity and reconstruction error (higher values of λ lead to more sparse solutions, at the cost of increased error). We denote this algorithm FOCUSS+, to distinguish from the FOCUSS algorithm [5] which omits the last line of (6). The estimate of $\mathbf{x}$ is refined over iterations of (6) and usually 10 to 50 iterations are needed for convergence (defined as the change in $\mathbf{x}$ being smaller than some threshold from one iteration to the next).

That the form of the nonnegative FOCUSS+ is closely related to FOCUSS is a fortunate property of the prior structure used here, and it is not the case in general that the nonnegative version of a sparse coding algorithm will be of similar form to the unrestricted version. The SBL-AVG algorithm of the next section is an example of a sparse coding algorithm that cannot easily be used for nonnegative coding.

2.2 Sparse Bayesian Learning with Adjustable-Variance Gaussian Priors (SBL-AVG)

Recently, a new class of Bayesian model characterized by Gaussian prior sources with adjustable variances has been developed [11]. These models use the linear generating model (1) for the data y but instead of using a non-Gaussian sparsity inducing prior on the sources x (as FOCUSS

does), they use a flexibly-parameterized Gaussian prior,

$$p_X(\mathbf{x}) = p(\mathbf{x}|\boldsymbol{\gamma}) = \prod_{i=0}^{n} \mathcal{N}(x_i|0, \gamma_i)\,, \tag{7}$$

where the variance hyperparameter γ_i can be adjusted for each component x_i. When γ_i approaches zero, the density of x_i becomes sharply peaked making it very likely that the source will be zero, increasing the sparsity of the code. The algorithm for estimating the sources has been termed *sparse Bayesian learning* (SBL), but we find this term to be too general, as other algorithms (including the earlier FOCUSS algorithm) also estimate sparse components in a Bayesian framework. We use the term SBL-AVG (adjustable-variance gaussian) to be more specific.

To insure that the prior probability $p(\mathbf{x}|\boldsymbol{\gamma})$ is sparsity-inducing, an appropriate prior on the hyperparameter $\boldsymbol{\gamma}$ must be chosen. In general, the Gamma$(\gamma_i^{-1}|a, b)$ distribution can be used for the prior of γ_i, and in particular with $a = b = 0$, the prior on γ_i becomes uniform. As shown in Section 3.2 of [26], this leads to $p(x_i)$ having a Student's t-distribution which qualitatively resembles the ℓ_p-norm-like distributions (with $0 < p < 1$) used to enforce sparsity in FOCUSS and other algorithms.

SBL-AVG has been used successfully for pattern recognition, with performance comparable to support vector machines (SVMs) [11, 26]. In these applications the known dictionary A is a kernel matrix created from the training examples in the pattern recognition problem just as with SVMs. The performance of SBL-AVG was similar to SVM in terms of error rates, while using far fewer support vectors (non-zero x_i) resulting in simpler models. Theoretical properties of SBL-AVG for subset selection have been elucidated in [23], and simulations on synthetic data show superior performance over FOCUSS and other basis selection methods. To our knowledge, results have not been previously reported for SBL-AVG on image coding.

The posterior density of $\mathbf{x}$ is a multivariate Gaussian,

$$p(\mathbf{x}|\mathbf{y}, \Gamma, \sigma^2) = \mathcal{N}(\boldsymbol{\mu}, \Sigma_{\mathbf{x}})\,, \tag{8}$$

which has mean and covariance,

$$\begin{aligned} \boldsymbol{\mu} &= \sigma^{-2}\Sigma_{\mathbf{x}} A^T \mathbf{y} \\ \Sigma_{\mathbf{x}} &= \left(\sigma^{-2} A^T A + \Gamma^{-1}\right)^{-1}\,, \end{aligned} \tag{9}$$

where the matrix Γ contains the hyperparameters γ_i, i.e. $\Gamma = \mathrm{diag}(\boldsymbol{\gamma})$. To implement the SBL-AVG algorithm for finding $\widehat{\mathbf{x}}$, we perform iterations of the update,

$$\begin{aligned} \widehat{\mathbf{x}} &\leftarrow \Gamma A^T (\sigma^2 I + A\Gamma A^T)^{-1}\mathbf{y} \\ \gamma_i &\leftarrow (\Sigma_{\mathbf{x}})_{i,i} + \mu_i^2\,. \end{aligned} \tag{10}$$

Iterative updates for the parameter σ^2 are given by,

$$\sigma^2 \leftarrow \frac{1}{m}\|\mathbf{y} - A\boldsymbol{\mu}\|^2 + \frac{\sigma^2}{m}\sum_{i=1}^{n}\left[1 - (\gamma_i^{-1}(\Sigma_{\mathbf{x}})_{i,i}\right]\,. \tag{11}$$

The iterations for the variance σ^2 and hyperparameters γ_i were derived in [23] using the expectation maximization (EM) algorithm and are assumed to be updated in parallel at each iteration. As the iterations proceed, some values of γ_i will be driven to 0, which leads to those components $x_i \to 0$, increasing the sparsity of the solution. For compression and coding applications, it is desirable to have a parameter that controls compression, and for SBL-AVG we use

a constant σ^2 (instead of the update for σ^2 in eq. 11). Higher values of σ^2 admit more error in the reconstruction, and so result in higher compression. Interestingly, the updates (10) are quite similar in form and computational complexity to the FOCUSS iterations (6) even though they are derived with different Bayesian priors, with the main difference being the update of the weighting matrices ($\Pi^{-1}(\widehat{\mathbf{x}})$ for FOCUSS and Γ for SBL-AVG). Software called "SparseBayes" that implements SBL-AVG can be found at `http://www.research.microsoft.com/mlp/RVM/default.htm`, which was used in the experiments below. Note that the algorithm in (10) is functionally equivalent to those presented in [11, 26] and that we have rewritten it to be consistent with our notation and emphasize the computational similarity to FOCUSS. However, creating a non-negative version of SBL-AVG proves much more difficult than for FOCUSS because of the need to integrate a Gaussian distribution with non-diagonal covariance over the positive orthant [27]. Naively adding a non-negative constraint to SBL-AVG (such as in the last line of eq. 6) does not result in a working algorithm.

2.3 Modified Matching Pursuit (MMP): Greedy vector selection

Many variations on the idea of matching pursuit, or greedy subset selection, have been developed [28, 29]. Here, we use modified matching pursuit (MMP) [12] which selects each vector (in series) to minimize the residual representation error. The simpler matching pursuit (MP) algorithm is more computationally efficient, but provides less accurate reconstruction. For the case of non-negative sources, matching pursuit can be suitably adapted, and we call this algorithm MP+.

In MMP, the maximum number of vectors to select, r, is prespecified. At each iteration $t = 1 \dots r$, a vector is added to the set of selected vectors $I_t = \{k_1 \dots k_t\}$,

$$k_t = \arg\max_l |\mathbf{a}_l^T \mathbf{b}_{t-1}| \, , \; l \notin I_{t-1} \, , \tag{12}$$

where $\mathbf{a}_l$ are the columns of A and $\mathbf{b}_{t-1}$ is the residual at iteration $t-1$. The selected vector at iteration t is denoted $\mathbf{a}_{k_t}$. For the first iteration, we set $\mathbf{b}_0 = \mathbf{y}$ (the signal to be represented). The residual is updated using,

$$\mathbf{b}_t = \mathbf{b}_{t-1} - (\mathbf{q}_t^T \mathbf{b}_{t-1}) \mathbf{q}_t \, , \tag{13}$$

where $\mathbf{q}_t$ is found by iteratively constructing an $\widehat{\mathbf{a}}_{k_t}^{(t)}$ as follows,

$$\begin{aligned} \widehat{\mathbf{a}}_{k_t}^{(0)} &= \mathbf{a}_{k_t} \, , \quad \mathbf{q}_0 = 0 \\ \widehat{\mathbf{a}}_{k_t}^{(i)} &= \widehat{\mathbf{a}}_{k_t}^{(i-1)} - \left(\mathbf{q}_{i-1}^T \, \widehat{\mathbf{a}}_{k_t}^{(i-1)} \right) \mathbf{q}_{i-1} \, , \quad i = 1 \dots t \\ \mathbf{q}_t &= \frac{\widehat{\mathbf{a}}_{k_t}^{(t)}}{\|\widehat{\mathbf{a}}_{k_t}^{(t)}\|} \, . \end{aligned} \tag{14}$$

The operation in (13) is a projection of the residual $\mathbf{b}_t$ onto the range space of the orthogonal complement of *all* the selected vectors. The simpler MP algorithm replaces the step (13) with a projection of the residual onto the orthogonal complement of the only the selected vector $\mathbf{a}_{k_t}$. The MP algorithm is more computationally efficient but provides less accurate reconstruction. More details and comparisons can be found in [12, 29].

The algorithm can be stopped when either r vectors have been chosen or when the residual is small enough, $\|\mathbf{b}_t\| \leq \epsilon$, where ϵ is a constant threshold that defines the maximum acceptable residual error. To find the coefficients of the selected vectors, a new matrix is created with the selected columns, $A_s = [\mathbf{a}_{k_1} \dots \mathbf{a}_{k_r}]$. The coefficient values corresponding to each vector are found using the pseudoinverse of A_s,

$$\mathbf{x}_s = (A_s^T A_s)^{-1} A_s^T \mathbf{y} \, . \tag{15}$$

To form the estimate $\widehat{\mathbf{x}}$, the elements of $\mathbf{x}_s$ are placed into the selected columns k_i.

3 Dictionary Learning Algorithms

In the previous section we discussed algorithms that accurately and sparsely represent a signal using a known, predefined dictionary A. Intuitively, we would expect that if A were adapted to the statistics of a particular problem that better and sparser representations could be found. This is the motivation that led to the development of the FOCUSS-CNDL dictionary learning algorithm. Dictionary learning is closely related to the problem of ICA which usually deals with a complete A but can be extended to an overcomplete A [6].

In this section we discuss the FOCUSS-CNDL [13] and overcomplete ICA algorithm of Lewicki and Sejnowski [6]. We briefly mention other overcomplete dictionary learning algorithms: Engan et al. [7, 30] developed the method of optimal directions (MOD) and applied it in an image compression system; Girolami [31] developed a variational approach similar to that of SBL-AVG; Palmer and Kreutz-Delgado [32] used the Bayesian maximum *a posteriori* (MAP) framework and a new notion of relative convexity to ensure sparse solutions; and Aharon et al. [33] developed an algorithm based on the singular value decomposition (K-SVD).

3.1 FOCUSS-CNDL

The FOCUSS-CNDL algorithm solves the problem (1) when both the sources $\mathbf{x}$ and the dictionary A are assumed to be unknown random variables [5]. The algorithm contains two major parts, a sparse vector selection step and a dictionary learning step which are derived in a jointly Bayesian framework. The sparse vector selection is done by FOCUSS (or FOCUSS+ if non-negative x_i are needed), and the dictionary learning A-update step uses gradient descent.

With a set of training data $Y = (\mathbf{y}_1, \ldots, \mathbf{y}_N)$ we find the maximum *a posteriori* estimates $\widehat{A}$ and $\widehat{X} = (\widehat{\mathbf{x}}_1, \ldots, \widehat{\mathbf{x}}_N)$ such that

$$(\widehat{A}, \widehat{X}) = \arg\min_{A,X} \sum_{k=1}^{N} \left[\|\mathbf{y}_k - A\mathbf{x}_k\|^2 + \lambda d_p(\mathbf{x}_k) \right] , \tag{16}$$

where $d_p(\mathbf{x}) = \|\mathbf{x}_k\|_p^p$ is the diversity measure (4) that measures (or approximates) the number of non-zero elements of a source vector $\mathbf{x}_k$ (see Section 2).

The optimization problem (16) attempts to minimize the squared error of the reconstruction of $\mathbf{y}_k$ while minimizing d_p and hence the number of non-zero elements in $\widehat{\mathbf{x}}_k$. The problem formulation is similar to ICA in that both model the input Y as being linearly generated by unknowns A and X, but ICA attempts to learn a new matrix W which linearly produces estimates $\widehat{\mathbf{x}}_k$ (by $W\mathbf{y}_k = \widehat{\mathbf{x}}_k$) where the components $\widehat{x}_{i,k}$ are as statistically independent as possible. ICA in general does not result in as sparse solutions as FOCUSS-CNDL which specifically uses the *sparsity-inducing* non-linear iterative FOCUSS algorithm to find $\widehat{\mathbf{x}}_k$.

We now summarize the FOCUSS-CNDL algorithm which was fully derived in [5]. For each of the N data vectors $\mathbf{y}_k$ in Y, we can update the sparse source vector $\widehat{\mathbf{x}}_k$ using one iteration of the FOCUSS or FOCUSS+ algorithm (6). After updating $\widehat{\mathbf{x}}_k$ for a certain number of the data vectors (the blocksize N_B) the dictionary $\widehat{A}$ is re-estimated,

$$\begin{aligned}
\Sigma_{\mathbf{y}\hat{\mathbf{x}}} &= \frac{1}{N_B} \sum_{k=1}^{N_B} \mathbf{y}_k \widehat{\mathbf{x}}_k^T , \quad \Sigma_{\hat{\mathbf{x}}\hat{\mathbf{x}}} = \frac{1}{N_B} \sum_{k=1}^{N_B} \widehat{\mathbf{x}}_k \widehat{\mathbf{x}}_k^T , \\
\delta\widehat{A} &= \widehat{A}\,\Sigma_{\hat{\mathbf{x}}\hat{\mathbf{x}}} - \Sigma_{\mathbf{y}\hat{\mathbf{x}}} \\
\widehat{A} &\leftarrow \widehat{A} - \eta \left(\delta\widehat{A} - \mathrm{tr}\,(\widehat{A}^T \delta\widehat{A})\widehat{A} \right) , \quad \gamma > 0 ,
\end{aligned} \tag{17}$$

where η is the learning rate parameter. Each iteration of FOCUSS-CNDL consists of updating all $\widehat{\mathbf{x}}_k, k = 1...N$ with one FOCUSS iteration (6), interspersed by dictionary updates (17) for every N_B vectors $\widehat{\mathbf{x}}_k$ (which uses Σ calculated from the updated $\widehat{\mathbf{x}}_k$ estimates). After each update of $\widehat{A}$, the columns are adjusted to have equal norm $\|\mathbf{a}_i\| = \|\mathbf{a}_j\|$, in such a way that $\widehat{A}$ has unit Frobenius norm, $\|\widehat{A}\|_F = 1$. Matlab code for the FOCUSS, FOCUSS-CNDL and non-negative variants can be found at `http://dsp.ucsd.edu/~jfmurray/software.htm`.

3.2 Overcomplete Independent Component Analysis (ICA)

Another method for learning an overcomplete dictionary based on ICA was developed by Lewicki and Sejnowski [14, 6]. In the overcomplete case, the sources must be estimated as opposed to in standard ICA (which assumes a complete dictionary A), where the sources are found by multiplying by a learned matrix W, yielding the estimates $\widehat{\mathbf{x}} = W\mathbf{y}$. In [14] the sources are estimated using a modified conjugate gradient optimization of a cost function closely related to (5) that uses the 1-norm (derived using a Laplacian prior on $\mathbf{x}$). The dictionary is updated by gradient ascent on the likelihood using a Gaussian approximation ([14], eq. 20).

Lewicki and Sejnowski treat the dictionary as a deterministic unknown and note that the classical maximum likelihood estimate of A is determined from maximizing the marginalized likelihood function,

$$p(Y|A) = \int p(Y, X|A)dX = \int p(Y|X, A)p(X)dX. \tag{18}$$

where $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ and $Y = (\mathbf{y}_1, \ldots, \mathbf{y}_N)$, and $\mathbf{x}_k$ and $\mathbf{y}_k$, $k = 1, \ldots, N$, are related via equation (1). Unfortunately, for supergaussian sparsity-inducing priors, such as the p-norm–like density shown in equations (3) and (4), this integration is generally intractable. To circumvent this problem Lewicki and Sejnowski approximate this integral by taking a Gaussian approximation to the prior evaluated at the MAP estimate of the source vectors X obtained from a current estimate of A. This is specifically done for the Laplacian $p = 1$ prior by solving the ℓ_1 (i.e., $p = 1$ basis pursuit) optimization problem using a conjugate gradient ℓ_1 optimization algorithm (see Section 3 of [14]).

After performing the marginalization integration, a dictionary update which "hill climbs" the resulting approximate likelihood function $\hat{p}(\widehat{X}|A)$ is given by,

$$\begin{aligned} \delta A &\leftarrow \hat{A}\left(\left\langle \mathbf{z}_k \hat{\mathbf{x}}_k^T \right\rangle_N + I\right) \\ \hat{A} &\leftarrow \hat{A} - \eta\, \delta A\,, \end{aligned} \tag{19}$$

where,

$$\mathbf{z}_k \triangleq \nabla_x \ln p(\hat{\mathbf{x}}_k)\,, \tag{20}$$

and $\langle\cdot\rangle_N$ denotes an N-sample average. The update rule (19) is valid for $p = 1$ as long as no single component $x_{k,i}$, $k = 1, \cdots, N$, $i = 1, \cdots n$, is identically zero. Using λ as in (3) for $p = 1$, the update rule (19) is equivalent to

$$\delta A \leftarrow \hat{A}\,(I - \lambda \Sigma^{\pi}_{\hat{\mathbf{x}}\hat{\mathbf{x}}}) \tag{21}$$

$$\hat{A} \leftarrow (1-\eta)\hat{A} + \lambda\eta \hat{A} \Sigma^{\pi}_{\hat{\mathbf{x}}\hat{\mathbf{x}}}\,, \tag{22}$$

where,

$$\Sigma^{\pi}_{\hat{\mathbf{x}}\hat{\mathbf{x}}} = \left\langle \Pi(\widehat{\mathbf{x}}_k)\, \hat{\mathbf{x}}_k \hat{\mathbf{x}}_k^T \right\rangle_N = \sum_{k=1}^{N} \Pi(\widehat{\mathbf{x}}_k)\, \hat{\mathbf{x}}_k \hat{\mathbf{x}}_k^T = \sum_{k=1}^{N} \mathrm{sign}(\hat{\mathbf{x}}_k) \hat{\mathbf{x}}_k^T\,, \tag{23}$$

with,

$$\Pi(\mathbf{x}) = \mathrm{diag}(|x_i|^{-1}) \quad \text{and} \quad \mathrm{sign}(\mathbf{x}) = [\mathrm{sign}(x_1), \cdots, \mathrm{sign}(x_n)]^T\,. \tag{24}$$

Matlab software for overcomplete ICA can be found at `http://www-2.cs.cmu.edu/~lewicki/`.

4 Measuring performance

To compare the performance of image coding algorithms we need to measure two quantities: distortion and compression. As a measure of distortion we use a normalized root-mean-square-error (RMSE) calculated over all N patches in the image,

$$\text{RMSE} = \frac{1}{\sigma}\left[\frac{1}{mN}\sum_{k=1}^{N}\|\mathbf{y}_k - A\widehat{\mathbf{x}}_k\|^2\right]^{\frac{1}{2}}, \tag{25}$$

where σ is the empirical estimate of the variance of the elements y_i (for all the $\mathbf{y}_k$, assuming i.i.d.), N is the number of image patches in the data set, and m is the size of each vector $\mathbf{y}_k$. Note that this is calculated over the image patches, leading to a slightly different calculation than the mean-square error over the entire image.

To measure how much a given transform algorithm compresses an image, we need a coding algorithm that maps which coefficients were used and their amplitudes into an efficient binary code. The design of such encoders is generally a complex undertaking, and is outside the scope of our work here. However, information theory states that we can estimate a lower bound on the coding efficiency if we know the entropy of the input signal. Following the method of Lewicki and Sejnowski (cf. [6] eq. 13) we estimate the entropy of the coding using histograms of the quantized coefficients. Each coefficient in $\widehat{\mathbf{x}}_k$ is quantized to 8 bits (or 256 histogram bins). The number of coefficients in each bin is c_i. The limit on the number of bits needed to encode each input vector is,

$$\#\text{bits} \geq \text{bits}_{\text{lim}} \equiv -\sum_{i=1}^{256}\frac{c_i}{N}\log_2 f_i\,, \tag{26}$$

where f_i is the estimated probability distribution at each bin. We use $f_i = c_i/(Nn)$, while in [6] a Laplacian kernel is used to estimate the density. The entropy estimate in bits/pixel is given by,

$$\text{entropy} = \frac{\text{bits}_{\text{lim}}}{m}\,, \tag{27}$$

where m is the size of each image patch (the vector $\mathbf{y}_k$). It is important to note that this estimate of entropy takes into account the extra bits needed to encode an overcomplete ($n > m$) dictionary, i.e. we are considering the bits used to encode each *image pixel*, not each coefficient.

5 Experiments

Previous work has shown that learned complete bases can provide more efficient image coding (fewer bits/pixel at the same error rate) when compared with unadapted bases such as Gabor, Fourier, Haar and Daubechies wavelets [14]. In our earlier work [5] we showed that overcomplete dictionaries A can give more efficient codes than complete bases. Here, our goal is to compare methods for learning overcomplete A (FOCUSS-CNDL and overcomplete ICA), and methods for coding images once A has been learned, including the case where the sources must be non-negative.

5.1 Comparison of dictionary learning methods

To provide a comparison between FOCUSS-CNDL and overcomplete ICA [6], both algorithms were used to train a 64×128 dictionary A on a set of 8×8 pixel patches drawn from images of man-made objects. For FOCUSS-CNDL, training of A proceeded as described in [5], for 150 iterations over $N = 20000$ image patches with the following parameters: learning rate $\eta = 0.01$, diversity

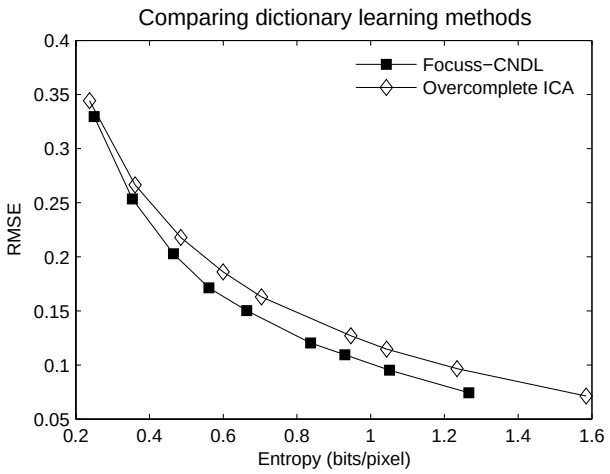

Figure 1: Image coding with 64x128 overcomplete dictionaries learned with FOCUSS-CNDL and overcomplete ICA. Images were sparsely coded using the FOCUSS algorithm with $p = 0.5$ and the compression level (bit rate) was adjusted by varying $\lambda_{\max} \in [0.005, 0.5]$, with higher values giving more compression (lower bit/pixel), left side of plot. Results are averaged over 15 images.

measure $p = 1.0$, blocksize $N_B = 200$, and regularization parameter $\lambda_{\max} = 2 \times 10^{-4}$. Training overcomplete ICA for image coding was performed as described in [14]. Both overcomplete ICA and FOCUSS-CNDL have many tunable parameters, and it is generally not possible to find the optimal values in the large parameter space. However, both algorithms have been tested extensively on image coding tasks. The parameters of overcomplete ICA used here were those in the implementation found at `http://www-2.cs.cmu.edu/~lewicki/`, which was shown by [14] to provide improved coding efficiency over non-learned bases (such as DCT and wavelet) as well as other learned bases (PCA and complete ICA). We believe that the parameters used have been sufficiently optimized for the image coding task to provide a reasonably fair comparison.

Once an A was learned with each method, FOCUSS was used to compare image coding performance, with parameters $p = 0.5$, iterations = 50, and the regularization parameter $\lambda_{\max}$ was adjusted over the range $[0.005, 0.5]$ to achieve different levels of compression (bits/pixel), with higher $\lambda_{\max}$ giving higher compression (lower bits/pixel). A separate test set was composed of 15 images of objects from the COIL database of rotated views of household objects [34].

Figure 1 shows the image coding performance of dictionaries learned using FOCUSS-CNDL and overcomplete ICA. Using the FOCUSS-CNDL dictionary provided better performance, i.e. at a given level of RMSE error images were encoded on average with fewer bits/pixel (bpp). FOCUSS was used to code the test images, which may give an advantage to the FOCUSS-CNDL dictionary as it was able to adapt its dictionary to sources generated with FOCUSS (while overcomplete ICA uses a conjugate gradient method to find sources).

5.2 Comparing image coding with MMP, SBL-AVG and FOCUSS

In this experiment we compare the coding performance of the MMP, SBL-AVG and FOCUSS vector selection algorithms using an overcomplete dictionary on a set of man-made images. The dictionary learned with FOCUSS-CNDL from the previous experiment was used, along with the same 15 test images. For FOCUSS, parameters were set as follows: $p = 0.5$, and compression (bits/pixel) was adjusted with $\lambda_{\max} \in [0.005, 0.5]$ as above. For SBL-AVG, we set the number

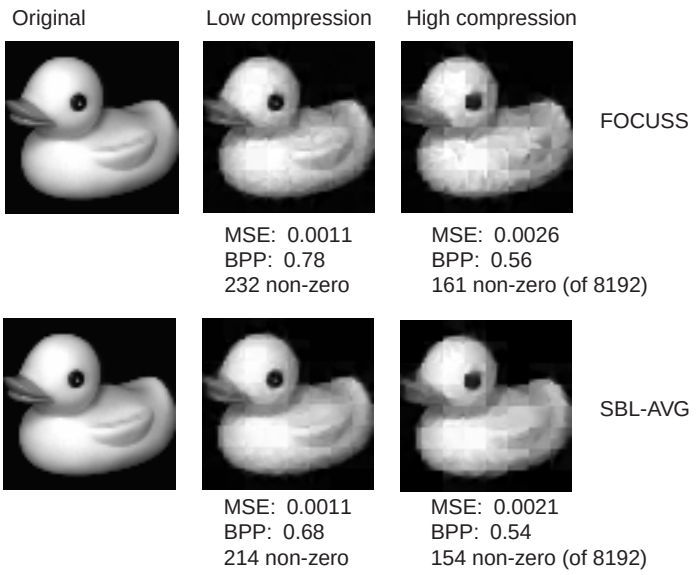

Figure 2: Images coded using an overcomplete dictionary (64x128) learned with FOCUSS-CNDL algorithm. Below each coded image are shown the mean-square error (MSE), the estimated entropy in bits/pixel (BPP) and the number of non-zero coefficients used to encode the entire image.

of iterations to 1000 and the constant noise parameter σ^2 was varied over $[0.005, 2.0]$ to adjust compression (with higher values of σ^2 giving higher compression). For MMP, the number of vectors selected r was varied from 1 to 13, with fewer vectors selected giving higher compression.

Figure 2 shows examples of an image coded with the FOCUSS and SBL-AVG algorithms. Images of size 64x64 pixels were coded at high and low compression levels. In both cases, SBL-AVG was more accurate and provided higher compression, e.g. MSE of 0.0021 vs. 0.0026 at entropy 0.54 vs 0.78 bits/pixel for the high compression case. In terms of sparsity, the SBL-AVG case in the bottom right of Figure 2 requires only 154 nonzero coefficients (of 8192, or about 2%) to represent the image.

Figure 3 shows the tradeoff between accurate reconstruction (low RMSE) and compression (bits/pixel) as approximated by the entropy estimate (27). The lower right of the curves represents the higher accuracy/lower compression regime, and in this range the SBL-AVG algorithm performs best, with lower RMSE error at the same level of compression. At the most sparse representation (upper left of the curves) where only 1 or 2 dictionary vectors are used to represent each image patch, the MMP algorithm performed best. This is expected in the case of 1 vector per patch, where the MMP finds the optimal single vector to match the input. Coding times per image on a 1.7 GHz AMD processor (Matlab implementation) are: FOCUSS 15.64 sec, SBL-AVG 17.96 sec, MMP 0.21 sec.

5.3 Image coding with non-negative sources

Next, we investigate the performance tradeoff associated with using non-negative sources $\mathbf{x}$. Using the same set of images as in the previous section, we learn a new $A \in \mathbb{R}^{64\times128}$ using the non-negative FOCUSS+ algorithm (6) in the FOCUSS-CNDL dictionary learning algorithm (17). The image gray-scale pixel values are scaled to $y_i \in [0, 1]$ and the sources are also restricted to $x_i \geq 0$ but elements of the dictionary are not further restricted and may be negative. Once the dictionary has been learned, the same set of 15 images as above were coded using FOCUSS+.

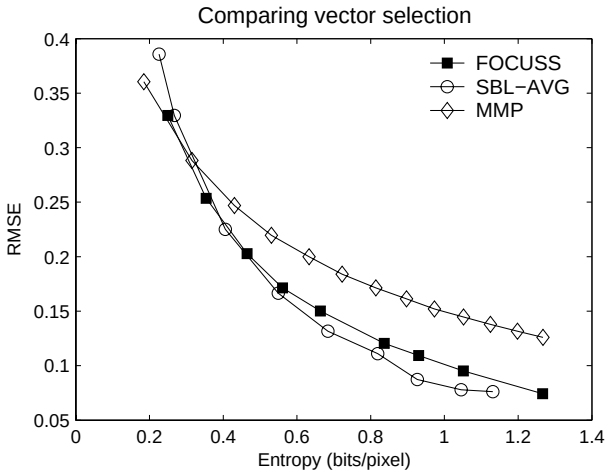

Figure 3: Comparison of sparse image coding algorithms with a 64x128 overcomplete dictionary. Compression rates are adjusted by varying parameters for each algorithm: $\lambda_{\max}$ for FOCUSS, σ^2 for SBL-AVG, and the number of vectors selected r for MMP. Results are averaged over 15 images.

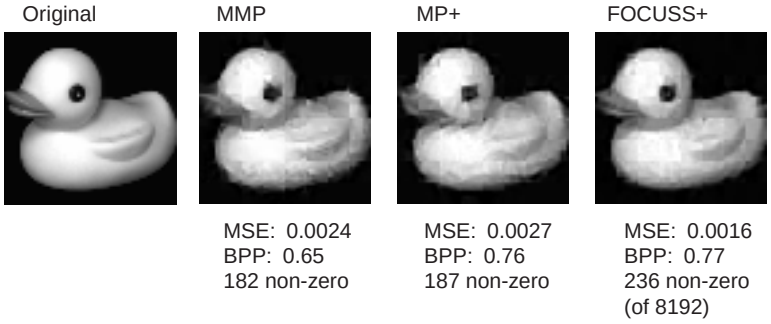

Figure 4: Image coding using non-negative sources (weights) with a 64x128 overcomplete dictionary learned with FOCUSS-CNDL+. Images were coded with MP+, FOCUSS+, and MMP (which uses negative coefficients, shown for comparison).

Figure 4 shows an image coded using MP+, FOCUSS+ and MMP (which uses negative coefficients). Restricting the coding to non-negative sources in MP+ shows relatively small increases in MSE and number of coefficients used, and a decrease in image quality. FOCUSS+ is visually superior and provides higher quality reconstruction (MSE 0.0016 vs. 0.0027) at comparable compression rates (0.77 vs. 0.76 bits/pixel). Figure 5 shows the compression/error tradeoff when using non-negative sources to code the same set of test images as above. As expected, there is a reduction in performance when compared with methods that use positive and negative sources especially at lower compression levels.

6 Potential for VLSI Implementation

While we have focused on the differences between the sparse coding and dictionary learning algorithms presented above, each may be suited to a particular class of application, which may require the use of dedicated VLSI or DSP hardware to achieve the needed speed and power

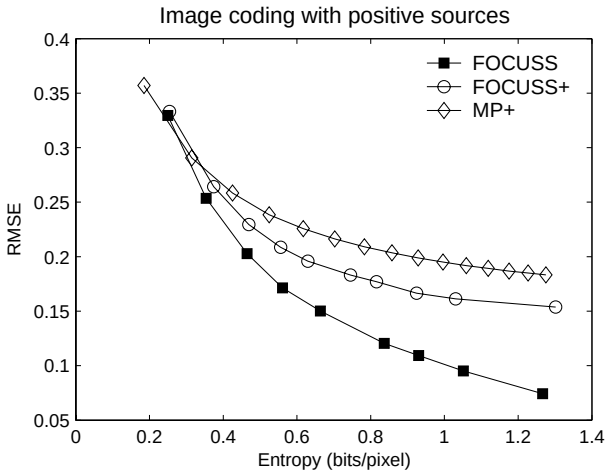

Figure 5: Image coding using non-negative sources x, with the FOCUSS curve from Figure 3 included for reference. Both experiments use a 64x128 overcomplete dictionary.

efficiency. From a VLSI implementation standpoint, all the algorithms share some desirable traits: they rely on easily parallelizable matrix operations, have simple logic flows (mainly repeated iterations), and have low memory requirements. For the sparse coding algorithms, the most time consuming operation is the matrix inversion required at each iterations (for MMP only one matrix inversion is required after the selected columns of A are chosen). Instead of computing the matrix inverse and subsequent matrix multiply in FOCUSS or SBL-AVG, the system of equations can be solved directly with Gaussian elimination [35]. Efficient parallel algorithms and architectures for Gaussian elimination have been developed, such as the division-free method of [36]. Progress continues to made in increasing the speed of the other required matrix algebra tasks, such as matrix multiplication [37, 38]. Using the algorithms of [36] and [37], we can find the number of multiplies and time-order required for each iteration of FOCUSS and SBL-AVG (Table 1). (See [36, 37] for details on architecture and number of processing elements required.)

For both the FOCUSS-CNDL and overcomplete ICA dictionary learning algorithms, the most time consuming step is the averaging of the sources in (17) and (23), which could be made more efficient with 2-D systolic arrays of processing elements [39]. For calculation of $\Sigma_{\hat{\mathbf{x}}\hat{\mathbf{x}}}$, an $n \times n$ array of multiply-add processing elements can perform the vector multiply and summation in one time step for each training sample $k \in 1 \ldots N$, reducing the time complexity from $O(Nn^2)$ for a serial implementation to $O(N)$. In FOCUSS-CNDL, a similar array of $m \times n$ elements is needed to find $\Sigma_{\mathbf{y}\hat{\mathbf{x}}}$.

7 Conclusion

We have discussed methods for finding sparse representations of images using overcomplete dictionaries, and methods for learning those dictionaries to be adapted to the problem domain. Images can be represented accurately with a very sparse code, with on the order of 2% of the coefficients being nonzero. When the sources are unrestricted, $\mathbf{x} \in \mathbb{R}^n$, the SBL-AVG algorithm provides the best performance, encoding images with fewer bits/pixel at the same error when compared FOCUSS and matching pursuit. When the sources are required to be non-negative, $x_i \geq 0$, the FOCUSS+ and associated dictionary learning algorithm presented

Table 1: Number of multiples required for certain steps of each iteration of the FOCUSS and SBL-AVG algorithms, given that the systems of equations in the second steps are solved with the Gaussian elimination algorithm of [36], and the matrix multiplies are performed using the algorithm of [37]. Time-order for these parallel algorithms is given in the right column.

FOCUSS (eq. 6)

Step of iteration	Multiplies	Time (parallel)
$\lambda I + A\Pi^{-1}A^T$	$nm + nm^2$	$m+1$
$\left(\lambda I + A\Pi^{-1}A^T\right)^{-1}\mathbf{y}$	$\frac{3}{4}(m^3+2m^2)+O(2m^2+m)$	$4m$
$\Pi^{-1}A^T\left(\lambda I + A\Pi^{-1}A^T\right)^{-1}\mathbf{y}$	nm^2	m
$\Pi^{-1} = \mathrm{diag}(\lvert\widehat{x}_i\rvert^{2-p})$	$O(n)$	1
Totals:	$\frac{3}{4}m^3+\frac{3}{2}m^2+2nm^2+nm+O(2m^2+m+n)$	$6m+2$

SBL-AVG (eq. 10)

Step of iteration	Multiplies	Time (parallel)
$\sigma^2 I + A\Gamma A^T$	$nm + nm^2$	$m+1$
$\Gamma A^T\left(\sigma^2 I + A\Gamma A^T\right)^{-1}$	$\frac{3}{4}(m^3+2nm^2)+O(2m^2+nm)$	$4m+n-1$
$\Gamma A^T\left(\sigma^2 I + A\Gamma A^T\right)^{-1}\mathbf{y}$	nm	1
$(\Sigma_{\mathbf{x}})_{i,i} = [\Gamma - \Gamma A^T\left(\sigma^2 I + A\Gamma A^T\right)^{-1}A\Gamma]_{i,i}$	nm	1
Totals:	$\frac{3}{4}m^3+\frac{5}{2}nm^2+3nm+O(2m^2+nm)$	$5m+n+2$

here provide the best performance. Based on the success of SBL-AVG, future work could include the development of dictionary learning algorithms that incorporate SBL-AVG into the vector selection step. While the increased performance of sparse overcomplete coding comes at the price of increased computational complexity, efficient parallel implementations in VLSI could make these algorithms more practical for many applications.

Acknowledgements

J.F. Murray gratefully acknowledges support from the Center for Magnetic Recording Research (CMRR) at UCSD, the Sloan Foundation and the ARCS Foundation. We also thank David Wipf for discussions regarding the SBL-AVG algorithm.

References

[1] R. C. Gonzales and R. E. Woods. *Digital Image Processing*. Addison-Wesley, Reading, MA, 1993.

[2] E. Oja. "A simplified neuron model as a principal component analyzer," *Journal of Mathematical Biology*, 15:267–273, 1982.

[3] T. T. Pham and R. J. P. deFigueiredo. "Maximum Likelihood Estimation of a Class of Non-Gaussian Densities with Application to ℓ_p Deconvolution," *IEEE Trans. Acoustics, Speech and Signal Processing*, 37(1):73–82, January 1989.

[4] C. Jutten and J. Hérault. "Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture," *Signal Processing*, 24:1–10, 1991.

[5] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T.-W. Lee, and T. J. Sejnowski. "Dictionary learning algorithms for sparse representation," *Neural Computation*, 15(2): 349–396, February 2003.

[6] M. S. Lewicki and T. J. Sejnowski. "Learning overcomplete representations," *Neural Computation*, 12(2):337–365, February 2000.

[7] K. Engan, J. H. Husoy, and S. O. Aase. "Frame based representation and compression of still images," In *Proc. ICIP 2001*, pages 1–4, 2001.

[8] B. A. Olshausen and D. J. Field. "Sparse coding with an overcomplete basis set: A strategy employed by V1?," *Vis. Res.*, 37:3311–3325, 1997.

[9] D. M. Weber and D. Casasent. "Quadratic Gabor filters for object detection," *IEEE Trans. Image Processing*, 10(2):218–230, February 2001.

[10] B. D. Rao and K. Kreutz-Delgado. "An affine scaling methodology for best basis selection," *IEEE Trans. Sig. Proc.*, 47:187–200, 1999.

[11] M. E. Tipping. "Sparse Bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, 1:211–244, 2001.

[12] S. F. Cotter, J. Adler, B. D. Rao, and K. Kreutz-Delgado. "Forward sequential algorithms for best basis selection," *IEE Proceedings Vision, Image and Signal Processing*, 146(5): 235–244, October 1999.

[13] J. F. Murray and K. Kreutz-Delgado. "An improved FOCUSS-based learning algorithm for solving sparse linear inverse problems," In *Conference Record of the 35th Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 347–351, Pacific Grove, CA, November 2001. IEEE.

[14] M. S. Lewicki and B. A. Olshausen. "A probabilistic framework for the adaptation and comparison of image codes," *J. Opt. Soc. Am. A*, 16(7):1587–1601, July 1999.

[15] K. Engan. *Frame based signal representation and compression*. PhD thesis, Stavanger Universty College, Norway, 2000.

[16] P. Paatero and U. Tapper. "Positive matrix factorization: A nonnegative factor model with optimal utilization of error estimates of data values," *Environmetrics*, 5:111–126, 1994.

[17] D. D. Lee and H. S. Seung. "Learning the parts of objects by non-negative matrix factorization," *Nature*, 401:788–791, October 1999.

[18] P. O. Hoyer. "Non-negative sparse coding," In *Proc. of the 12th IEEE Workshop on Neural Networks for Sig. Proc.*, pages 557– 565, 2002.

[19] M. D. Plumbley. "Algorithms for nonnegative independent component analysis," *IEEE Trans. Neural Net.*, 14(3):534–543, May 2003.

[20] S. Chen and D. Donoho. "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

[21] D. Donoho and M. Elad. "Optimally sparse representation in general (nonorthogonal) dictionaries via l_1 minimization," *Proceedings of the National Academy of Sciences*, 100(5), March 2003.

[22] D. P. Wipf and B. D. Rao. "Probabilistic analysis for basis selection via ℓ_p diversity measures," In *Proc. of the Intl. Conf. on Acoustics, Speech, and Signal Processing, 2004 (ICASSP '04)*, volume 2, May 2004.

[23] D. P. Wipf and B. D. Rao. "Sparse Bayesian learning for basis selection," *IEEE Transactions on Signal Processing*, 52(8):2153–2164, 2004.

[24] I. F. Gorodnitsky, J. S. George, and B. D. Rao. "Neuromagnetic source imaging with FOCUSS: a recursive weighted minimum norm algorithm," *Electroencephalography and Clinical Neurophysiology*, 95(4):231–251, 1995.

[25] R. Vigário and E. Oja. "Independence: A new criterion for the analysis of the electromagnetic fields in the global brain?," *Neural Networks*, 13:891–907, 2000.

[26] C. M. Bishop and M. E. Tipping. *Advances in Learning Theory: Methods, Models and Applications*, volume 190 of *NATO Science Series III: Computer and Systems Sciences*, chapter Bayesian regression and classification, pages 267–285. IOS Press, 2003.

[27] R. J. Muirhead. *Aspects of Multivariate Statistical Theory*. Wiley-Interscience, 1982.

[28] S. G. Mallat and Z. Zhang. "Matching Pursuits with Time-Frequency Dictionaries," *IEEE Trans. Sig. Proc.*, 41(12):3397–3415, 1993.

[29] S. F. Cotter. *Subset selection algorithms with applications*. PhD thesis, Univ. of California at San Diego, 2001.

[30] K. Engan, K. Skretting, and J. H. Husoy. "A family of iterative LS-based dictionary learning algorithms, ILS-DLA, for sparse signal represenation," *submitted*, 2005.

[31] M. Girolami. "A variational method for learning sparse and overcomplete representations," *Neural Computation*, 13:2517–2532, 2001.

[32] J. A. Palmer and K. Kreutz-Delgado. "A general framework for component estimation," In *Proc. 4th Intl. Symp. on ICA*, 2003.

[33] M. Aharon, M. Elad, and A. Bruckstein. "K-SVD: an algorithm for designining of overcomplete dictionaries for sparse representation," *submitted*, 2005.

[34] S. A. Nene, S. K. Nayar, and H. Murase. "Columbia object image library (COIL-100)," Technical Report CUCS-006-96, Columbia University, 1996.

[35] G. H. Golub and C. F. V. Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, MD, 1983.

[36] S. Peng and S. Sedukhin. "Parallel algorithm and architectures for two-step division-free Gaussian elimination," In *Algorithms and Architectures for Parallel Processing, 3rd International Conference on (ICAPP 97)*, pages 489–502, 1997.

[37] J.-C. Tsay and P.-Y. Chang. "Design of efficient regular arrays for matrix multiplication by two-step regularization," *IEEE Transactions on Parallel and Distributed Systems*, 6(2): 215–222, February 1995.

[38] K. Muhammad and K. Roy. "Reduced computational redundancy implementation of DSP algorithms using computation sharing vector scaling," *IEEE Transactions on VLSI Systems*, 10(3):292–300, 2002.

[39] D. Zhang. *Parallel VLSI Neural System Design*. Springer-Verlag, Singapore, 1998.