

Dictionary Learning for L1-Exact Sparse Coding

Mark D. Plumbley

Department of Electronic Engineering, Queen Mary University of London,
Mile End Road, London E1 4NS, United Kingdom.
Email: `mark.plumbley@elec.qmul.ac.uk`

Abstract. We have derived a new algorithm for dictionary learning for sparse coding in the ℓ_1 exact sparse framework. The algorithm does not rely on an approximation residual to operate, but rather uses the special geometry of the ℓ_1 exact sparse solution to give a computationally simple yet conceptually interesting algorithm. A self-normalizing version of the algorithm is also derived, which uses negative feedback to ensure that basis vectors converge to unit norm. The operation of the algorithm is illustrated on a simple numerical example.

1 Introduction

Suppose we have a sequence of observations $\mathbf{X} = [\mathbf{x}^1, \dots, \mathbf{x}^p]$, $\mathbf{x}^k \in \mathbb{R}^n$. In the sparse coding problem [1] we wish to find a dictionary matrix \mathbf{A} and representation matrix \mathbf{S} such that

$$\mathbf{X} = \mathbf{A}\mathbf{S} \quad (1)$$

and where the representations $\mathbf{s}^k \in \mathbb{R}^m$ in the matrix $\mathbf{S} = [\mathbf{s}^1, \dots, \mathbf{s}^p]$ are *sparse*, i.e. where there are few non-zero entries in each \mathbf{s}_j . In the case where we look for solutions $\mathbf{X} = \mathbf{A}\mathbf{S}$ with no error, we say that this is an *exact sparse* solution. In the sparse coding problem we typically have $m > n$, so this is closely related to the overcomplete independent component analysis (overcomplete ICA) problem, which has the additional assumption that the components of the representation vectors \mathbf{s}_j are statistically independent.

If the dictionary \mathbf{A} is given, then for each sample $\mathbf{x} = \mathbf{x}^k$ we can separately look the sparsest representation

$$\min_{\mathbf{s}} \|\mathbf{s}\|_0 \quad \text{such that} \quad \mathbf{x} = \mathbf{A}\mathbf{s}. \quad (2)$$

However, even this is a hard problem, so one approach is to solve instead the ‘relaxed’ ℓ_1 -norm problem

$$\min_{\mathbf{s}} \|\mathbf{s}\|_1 \quad \text{such that} \quad \mathbf{x} = \mathbf{A}\mathbf{s}. \quad (3)$$

This approach, known in the signal processing literature as Basis Pursuit [2], can be solved efficiently with linear programming methods (see also [3]).

Even if such efficient sparse representation methods exist, learning the dictionary \mathbf{A} is a non-trivial task. Several methods have been proposed in the

literature, such as those by Olshausen and Field [4] and Lewicki and Sejnowski [5], and these can be derived within a principled probabilistic framework [1]. A recent alternative is the K-SVD algorithm [6], which is a generalization of the K-means algorithm.

However, many of these algorithms are designed to solve the sparse approximation problem $\mathbf{X} = \mathbf{A}\mathbf{S} + \mathbf{R}$ for some nonzero residual term \mathbf{R} , rather than the exact sparse problem (1). For example, the Olshausen and Field [4] approximate maximum likelihood algorithm is

$$\Delta\mathbf{A} = \eta E(\mathbf{r}\mathbf{s}^T) \quad (4)$$

where $\mathbf{r} = \mathbf{x} - \mathbf{A}\mathbf{s}$ is the residual after approximation, and the K-SVD algorithm [1] minimizes the norm $\|\mathbf{R}\|_F = \|\mathbf{X} - \mathbf{A}\mathbf{S}\|_F$. If we have a sparse representation algorithm that is successfully able to solve (3) exactly on each data sample \mathbf{x}^k , then we have a zero residual $\mathbf{R} = \mathbf{0}$, and there is nothing to ‘drive’ the dictionary learning algorithm. Some other dictionary learning algorithms have other constraints: for example, the method of Georgiev et al [7] requires at most $m - 1$ nonzero elements in each column of \mathbf{S} .

While these algorithms have been successful for practical problems, in this paper we specifically explore the special geometry of the ℓ_1 exact sparse dictionary learning problem. We shall derive a new dictionary learning algorithm for the ℓ_1 exact sparse problem, using the basis vertex $\mathbf{c} = (\overline{\mathbf{A}}^\dagger)^T \mathbf{1}$ associated with a subdictionary (basis set) $\overline{\mathbf{A}}$ identified in the ℓ_1 exact sparse representation problem (3).

2 The Dual Problem and Basis Vertex

The linear program (3) has a corresponding *dual* linear program [2]

$$\max_{\mathbf{c}} \mathbf{x}^T \mathbf{c} \quad \text{such that} \quad \pm \mathbf{a}_j^T \mathbf{c} \leq 1 \quad j = 1, \dots, m \quad (5)$$

which has an optimum \mathbf{c}^* associated with any optimum \mathbf{s}^* of (3). In a previous paper we explored the polytope geometry of this type of dual problem, and derived an algorithm, Polytope Faces Pursuit (PFP), which searches for the optimal vertex which maximizes $\mathbf{x}^T \mathbf{c}$, and uses that to find the optimal vector \mathbf{s} [8]. Polytope Faces Pursuit is a gradient projection method [9] which iteratively builds a solution basis $\overline{\mathbf{A}}$ consisting of a subset of the signed columns $\sigma_j \mathbf{a}_j$ of \mathbf{A} , $\sigma_j \in \{-1, 0, +1\}$, chosen such that $\mathbf{x} = \overline{\mathbf{A}}\overline{\mathbf{s}}$ with $\overline{\mathbf{s}} > \mathbf{0}$ containing the absolute value of the nonzero coefficients of \mathbf{s} at the solution. The algorithm is similar in structure to orthogonal matching pursuit (OMP), but with a modified admission criterion

$$\mathbf{a}' = \arg \max_{\mathbf{a}_i} \frac{\mathbf{a}_i^T \mathbf{r}}{1 - \mathbf{a}_i^T \mathbf{c}} \quad (6)$$

to add a new basis vector \mathbf{a}' to the current basis set, together with an additional rule to switch out basis vectors which are no longer feasible.

The basis vertex $\mathbf{c} = (\overline{\mathbf{A}}^\dagger)^T \mathbf{1}$ is the solution to the dual problem (5). During the operation of the algorithm \mathbf{c} satisfies $\mathbf{A}^T \mathbf{c} \leq \mathbf{1}$, so it remains *dual-feasible* throughout. For all active atoms \mathbf{a}_j in the current basis set, we have $\mathbf{a}_j^T \mathbf{c} = 1$. Therefore at the minimum ℓ_1 norm solution the following conditions hold:

$$\overline{\mathbf{A}}^T \mathbf{c} = \mathbf{1} \quad (7)$$

$$\mathbf{x} = \overline{\mathbf{A}} \overline{\mathbf{s}} \quad \overline{\mathbf{s}} > \mathbf{0}. \quad (8)$$

We will use these conditions in our derivation of the dictionary learning algorithm that follows.

3 Dictionary Learning Algorithm

We would like to construct an algorithm to find the matrix \mathbf{A} that minimizes the total ℓ_1 norm

$$J = \sum_{k=1}^p \|\mathbf{s}^k\|_1 \quad (9)$$

where \mathbf{s}^k is chosen such that $\mathbf{x}^k = \mathbf{A} \mathbf{s}^k$ for all $k = 1, \dots, p$, i.e. such that $\mathbf{X} = \mathbf{A} \mathbf{S}$, and where the columns of \mathbf{A} are constrained to have unit norm. In particular, we would like to construct an iterative algorithm to adjust \mathbf{A} to reduce the total ℓ_1 norm (9): let us therefore investigate how J depends on \mathbf{A} .

For the contribution due to the k th sample we have $J = \sum_k J^k$ where $J^k = \|\mathbf{s}^k\|_1 = \mathbf{1}^T \overline{\mathbf{s}}^k$ since $\overline{\mathbf{s}}^k \geq 0$. Dropping the superscripts k from \mathbf{x}^k , $\overline{\mathbf{A}}^k$ and $\overline{\mathbf{s}}^k$ we therefore wish to find how $J^k = \mathbf{1}^T \overline{\mathbf{s}}$ changes with $\overline{\mathbf{A}}$, so taking derivatives of J^k we get

$$dJ^k/dt = \mathbf{1}^T (d\overline{\mathbf{s}}/dt). \quad (10)$$

Now taking the derivative of (8) for fixed \mathbf{x} we get

$$0 = (d\overline{\mathbf{A}}/dt) \overline{\mathbf{s}} + \overline{\mathbf{A}} (d\overline{\mathbf{s}}/dt) \quad (11)$$

and pre-multiplying by \mathbf{c}^T gives us

$$\mathbf{c}^T (d\overline{\mathbf{A}}/dt) \overline{\mathbf{s}} = -\mathbf{c}^T \overline{\mathbf{A}} (d\overline{\mathbf{s}}/dt) \quad (12)$$

$$= -\mathbf{1}^T (d\overline{\mathbf{s}}/dt) \quad (13)$$

$$= -dJ^k/dt \quad (14)$$

where the last two equations follow from (7) and (10). Introducing $\text{trace}(\cdot)$ for the trace of a matrix, we can rearrange this to get

$$\frac{dJ^k}{dt} = -\text{trace} \left(\mathbf{c}^T \frac{d\overline{\mathbf{A}}}{dt} \overline{\mathbf{s}} \right) = -\text{trace} \left((\mathbf{c} \overline{\mathbf{s}}^T)^T \frac{d\overline{\mathbf{A}}}{dt} \right) = -\left\langle \mathbf{c} \overline{\mathbf{s}}^T, \frac{d\overline{\mathbf{A}}}{dt} \right\rangle \quad (15)$$

from which we see that the gradient of J^k with respect to $\overline{\mathbf{A}}$ is given by $\nabla_{\overline{\mathbf{A}}} J^k = -\mathbf{c} \overline{\mathbf{s}}^T$. Summing up over all k and applying to the original matrix \mathbf{A} we get

$$\nabla_{\mathbf{A}} J = -\sum_k \mathbf{c}^k (\mathbf{s}^k)^T = -\mathbf{C} \mathbf{S}^T \quad (16)$$

with $\mathbf{C} = [\mathbf{c}^k]$, a somewhat surprisingly simple result. Therefore the update

$$\Delta \mathbf{A} = \eta \sum_k \mathbf{c}^k (s_j^k)^T \quad (17)$$

will perform a steepest descent search for the minimum total ℓ_1 norm J , and any path $d\mathbf{A}/dt$ for which $\langle (d\mathbf{A}/dt), \nabla_{\mathbf{A}} J \rangle < 0$ will cause J to decrease.

3.1 Unit norm atom constraint

Now without any constraint, algorithm (17) will tend to reduce the ℓ_1 norm by causing \mathbf{A} to increase without bound, so we need to impose a constraint on \mathbf{A} . A common constraint is to require the columns \mathbf{a}_j of \mathbf{A} to be unit vectors, $\|\mathbf{a}_j\|_2^2 = 1$, i.e. $\mathbf{a}_j^T \mathbf{a}_j = 1$. We therefore require our update to be restricted to paths $d\mathbf{a}_j/dt$ for which $\mathbf{a}_j^T (d\mathbf{a}_j/dt) = 0$.

To find the projection of (16) in this direction, consider the gradient component

$$\mathbf{g}_j = \frac{dJ}{d\mathbf{a}_j} = - \sum_k \mathbf{c}^k s_j^k. \quad (18)$$

The orthogonal projection of \mathbf{g}_j onto the required tangent space is given by

$$\tilde{\mathbf{g}}_j = \mathbf{P}_{\mathbf{a}_j} \mathbf{g}_j = \left(\mathbf{I} - \frac{\mathbf{a}_j \mathbf{a}_j^T}{\|\mathbf{a}_j\|_2^2} \right) \mathbf{g}_j = \mathbf{g}_j - \frac{1}{\|\mathbf{a}_j\|_2^2} \mathbf{a}_j (\mathbf{a}_j^T \mathbf{g}_j). \quad (19)$$

Now considering the rightmost factor $\mathbf{a}_j^T \mathbf{g}_j$, from (18) we get

$$\mathbf{a}_j^T \mathbf{g}_j = - \sum_k \mathbf{a}_j^T \mathbf{c}^k s_j^k. \quad (20)$$

Considering just the k th term $\mathbf{a}_j^T \mathbf{c}^k s_j^k$, if \mathbf{a}_j is one of the basis vectors in $\overline{\mathbf{A}}^k$ (with possible change of sign $\sigma_j^k = \text{sign}(s_j^k)$) which forms part of the solution $\mathbf{x}^k = \overline{\mathbf{A}}^k \overline{\mathbf{s}}^k$ found by a minimum ℓ_1 norm solution, then we must have $\sigma_j^k \mathbf{a}_j^T \mathbf{c}^k = 1$ where $\sigma_j^k = \text{sign}(s_j^k)$, because $\overline{\mathbf{A}}^{kT} \mathbf{c} = \mathbf{1}$, so $\mathbf{a}_j^T \mathbf{c}^k s_j^k = \sigma_j^k s_j^k = |s_j^k|$. On the other hand, if \mathbf{a}_j does not form part of $\overline{\mathbf{A}}^k$, then $s_j^k = 0$ so $\mathbf{a}_j^T \mathbf{c}^k s_j^k = 0 = |s_j^k|$. Thus regardless of the involvement of \mathbf{a}_j in $\overline{\mathbf{A}}^k$, we have $\mathbf{a}_j^T \mathbf{c}^k s_j^k = |s_j^k|$, so

$$\mathbf{a}_j^T \mathbf{g}_j = - \sum_k |s_j^k| \quad (21)$$

and therefore

$$\tilde{\mathbf{g}}_j = - \sum_k \left(\mathbf{c}^k s_j^k - \frac{1}{\|\mathbf{a}_j\|_2^2} \mathbf{a}_j |s_j^k| \right). \quad (22)$$

Therefore we have the following ‘tangent’ update rule:

$$\mathbf{a}_j(T+1) = \mathbf{a}_j(T) + \eta \sum_k \left(\mathbf{c}^k s_j^k - \frac{1}{\|\mathbf{a}_j(T)\|_2^2} \mathbf{a}_j(T) |s_j^k| \right) \quad (23)$$

which will perform a tangent-constrained steepest descent update to find the minimum total ℓ_1 norm J . We should note that the tangent update is not entirely sufficient to constrain \mathbf{a}_j to remain of unit norm, so an occasional renormalization step $\mathbf{a}_j \leftarrow \mathbf{a}_j / \|\mathbf{a}_j\|_2$ will be required after a number of applications of (23).

4 Self-normalizing Algorithm

Based on the well-known negative feedback structure used in PCA algorithms such as the Oja [10] PCA neuron, we can modify algorithm (23) to produce the following self-normalizing algorithm that does not require the explicit renormalization step:

$$\mathbf{a}_j(T+1) = \mathbf{a}_j(T) + \eta \sum_k (\mathbf{c}^k s_j^k - \mathbf{a}_j(T) |s_j^k|) \quad (24)$$

where we have simply removed the factor $1/\|\mathbf{a}_j(T)\|_2^2$ from the second term in (23). This algorithm is computationally very simple, and suggests an online version $\mathbf{a}_j(k) = \mathbf{a}_j(k-1) + \eta (\mathbf{c}^k s_j^k - \mathbf{a}_j(k-1) |s_j^k|)$ with the dictionary updated as each data point is presented.

For unit norm basis vectors $\|\mathbf{a}_j(T)\|_2 = 1$, the update produced by algorithm (24) is identical to that produced by the tangent algorithm (23). Therefore, for unit norm basis vectors, algorithm (24) produces a step in a direction which reduces J . (Note that algorithm (24) will not necessarily reduce J when \mathbf{a}_j is not unit norm.)

To show that the norm of the basis vectors \mathbf{a}_j in algorithm (24) converge to unit length, we require that each \mathbf{a}_j must be involved in the representation of at least one pattern \mathbf{x}^k , i.e. for some k we have $s_j^k \neq 0$. (If this were not true, that basis vector would have been ignored completely so would not be updated by the algorithm.) Consider the ordinary differential equation (ode) version of (24):

$$\frac{d\mathbf{a}_j}{dt} = \sum_k (\mathbf{c}^k s_j^k - \mathbf{a}_j |s_j^k|) \quad (25)$$

$$= -\tilde{\mathbf{g}}_j + \left(\frac{1}{\|\mathbf{a}_j\|_2^2} - 1 \right) \mathbf{a}_j \sum_k |s_j^k| \quad (26)$$

$$= -\tilde{\mathbf{g}}_j + \frac{1}{\|\mathbf{a}_j\|_2^2} \left(1 - \|\mathbf{a}_j\|_2^2 \right) \mathbf{a}_j \sum_k |s_j^k| \quad (27)$$

which, noting that $\mathbf{a}_j^T \tilde{\mathbf{g}}_j = \mathbf{a}_j^T \mathbf{P}_{\bar{\mathbf{a}}_j} \mathbf{g}_j = 0$, gives us

$$\mathbf{a}_j^T \frac{d\mathbf{a}_j}{dt} = \frac{1}{\|\mathbf{a}_j\|_2^2} \left(1 - \|\mathbf{a}_j\|_2^2 \right) \mathbf{a}_j^T \mathbf{a}_j \sum_k |s_j^k| = \left(1 - \|\mathbf{a}_j\|_2^2 \right) \sum_k |s_j^k|. \quad (28)$$

Constructing the Lyapunov function [11] $Q = (1/4)(1 - \|\mathbf{a}_j\|_2^2)^2 \geq 0$, which is zero if and only if \mathbf{a}_j has unit length, we get

$$dQ/dt = -(1 - \|\mathbf{a}_j\|_2^2)\mathbf{a}_j^T(d\mathbf{a}_j/dt) \quad (29)$$

$$= -(1 - \|\mathbf{a}_j\|_2^2)^2 \sum_k |s_j^k| \quad (30)$$

$$\leq 0 \quad (31)$$

where $\sum_k |s_j^k| > 0$ since at least one of the s_j^k is nonzero, so equality holds in (31) if and only if $\|\mathbf{a}_j\|_2^2 = 0$. Therefore the ode (25) will cause $\|\mathbf{a}_j\|_2^2$ to converge to 1 for all basis vectors \mathbf{a}_j .

While algorithm (24) does not strictly require renormalization, we found experimentally that an explicit unit norm renormalization step did produce slightly more consistent behaviour in reduction of the total ℓ_1 norm J .

Finally we note that at convergence of algorithm (24), the basis vectors must satisfy

$$\mathbf{a}_j = \frac{\sum_k \mathbf{c}^k s_j^k}{\sum_k |s_j^k|} \quad (32)$$

so that \mathbf{a}_j must be a (signed) weighted sum of the basis vertices \mathbf{c}^k in which it is involved. While equation (32) is suggestive of a fixed point algorithm, we have observed that it yields unstable behaviour if used directly. Nevertheless we believe that it would be interesting to explore this in future for the final stages of an algorithm, as it nears convergence.

5 Augmenting Polytope Faces Pursuit

After an update to the dictionary \mathbf{A} , it is not necessary to restart the search for the minimum ℓ_1 norm solutions \mathbf{s}^k to $\mathbf{x}^k = \mathbf{A}\mathbf{s}^k$ from $\mathbf{s}^k = \mathbf{0}$. In many cases the dictionary vector will have changed only slightly, so the signs σ_j^k and selected subdictionary $\overline{\mathbf{A}}^k$ may be very similar to the previous solution, before the dictionary update. At the T th dictionary learning step we can therefore restart the search for $\mathbf{s}^k(T)$ from the basis set selected by the last solution $\mathbf{s}^k(T-1)$.

However, if we start from the same subdictionary selection pattern after a change to the dictionary, we can no longer guarantee that the solution will be *dual-feasible*, i.e. that (5) is always satisfied, which is required for the Polytope Faces Pursuit algorithm [8]. While we will still have $\mathbf{a}_j^T \mathbf{c} = 1$ for all vectors \mathbf{a}_j in the solution subdictionary, we may have increased some other \mathbf{a}_j , not in the original solution set, so that now $\mathbf{a}_j^T \mathbf{c} > 1$.

To overcome this problem, if $\mathbf{a}_j^T \mathbf{c}^k > 1$ such that dual feasibility fails for a particular sample k and basis vector \mathbf{a}_j , we simply restart the sparse Polytope Faces Pursuit algorithm from $\mathbf{s}^k = \mathbf{0}$ for this particular sample, to guarantee that dual-feasibility is restored. We believe that it may be possible to construct

a more efficient method to restore dual-feasibility, based on selectively swapping vectors to bring the solution into feasibility, but it appears to be non-trivial to guarantee that loops will not result.

6 Numerical Illustration

To illustrate the operation of the algorithm, Figure 1 shows a small graphical example. Here four source variables s_j are generated with identical mixture-of-

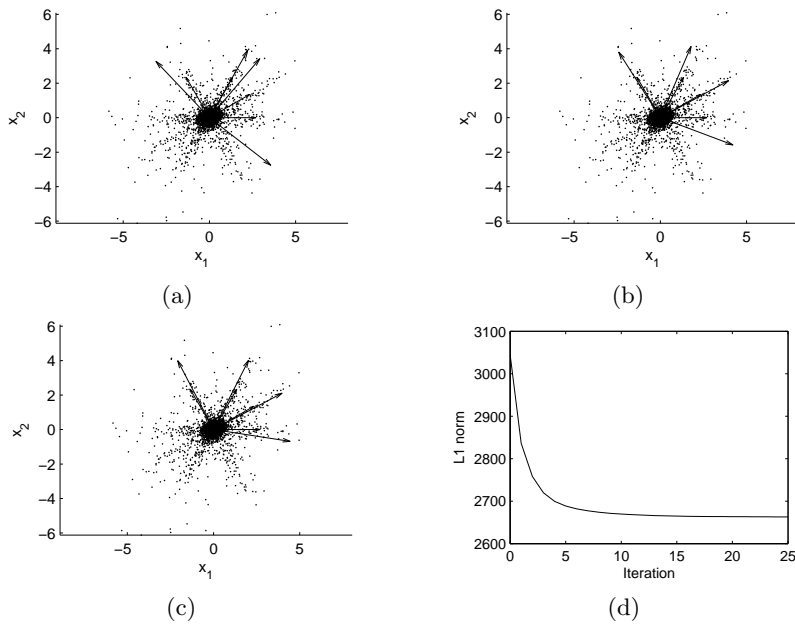


Fig. 1. Illustration of dictionary learning of a $n = 2$ dimensional mixture of $m = 4$ MoG-distributed sources, for $p = 3000$ samples. The plots show (a) the initial condition, and updates after (b) 5 dictionary updates and (c) 25 dictionary updates. The learning curve (d) confirms that the ℓ_1 norm decreases as the algorithm progresses. On (a)-(c), the longer arrows are scaled versions of the learned dictionary vectors \mathbf{a}_j , with the shorter arrows showing the directions of the generating dictionary vectors for comparison.

gaussian (MoG) densities in an $n = 2$ dimensional space and added with angles $\theta \in \{0, \pi/6, \pi/3, 4\pi/6\}$. It is important to note that, even in the initial condition Figure 1(a), the basis set spans the input space and optimization of (3) has an exact solution $\mathbf{x}^k = \mathbf{A}\mathbf{s}^k$ for all data samples \mathbf{x}^k , at least to within numerical precision of the algorithm. Therefore this situation would not be suitable for any dictionary learning algorithm which relies on a residual $\mathbf{r} = \mathbf{x} - \mathbf{A}\mathbf{s}$.

7 Conclusions

We have derived a new algorithm for dictionary learning for sparse coding in the ℓ_1 exact sparse framework. The algorithm does not rely on an approximation residual to operate, but rather uses the special geometry of the ℓ_1 exact sparse solution to give a computationally simple yet conceptually interesting algorithm. A self-normalizing version of the algorithm is also derived, which uses negative feedback to ensure that basis vectors converge to unit norm.

The operation of the algorithm was illustrated on a simple numerical example. While we emphasize the derivation and geometry of the algorithm in the present paper, we are currently working on applying this new algorithm to practical sparse approximation problems, and will present these results in future work.

8 Acknowledgements

This work was supported by EPSRC grants GR/S75802/01, GR/S82213/01, GR/S85900/01, EP/C005554/1 and EP/D000246/1.

References

1. Kreutz-Delgado, K., Murray, J.F., Rao, B.D., Engan, K., Lee, T.W., Sejnowski, T.J.: Dictionary learning algorithms for sparse representation. *Neural Computation* **15** (2003) 349–396
2. Chen, S.S., Donoho, D.L., Saunders, M.A.: Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing* **20** (1998) 33–61
3. Bofill, P., Zibulevsky, M.: Underdetermined blind source separation using sparse representations. *Signal Processing* **81** (2001) 2353–2362
4. Olshausen, B.A., Field, D.J.: Emergence of simple-cell receptive-field properties by learning a sparse code for natural images. *Nature* **381** (1996) 607–609
5. Lewicki, M.S., Sejnowski, T.J.: Learning overcomplete representations. *Neural Computation* **12** (2000) 337–365
6. Aharon, M., Elad, M., Bruckstein, A.: K-SVD: Design of dictionaries for sparse representation. In: *Proceedings of SPARS'05, Rennes, France.* (2005) 9–12
7. Georgiev, P., Theis, F., Cichocki, A.: Sparse component analysis and blind source separation of underdetermined mixtures. *IEEE Transactions on Neural Networks* **16** (2005) 992–996
8. Plumbley, M.D.: Recovery of sparse representations by polytope faces pursuit. In: *Proceedings of the 6th International Conference on Independent Component Analysis and Blind Source Separation (ICA 2006), Charleston, SC, USA.* (2006) 206–213 LNCS 3889.
9. Rosen, J.B.: The gradient projection method for nonlinear programming. Part I. Linear constraints. *J. Soc. Indust. Appl. Math.* **8** (1960) 181–217
10. Oja, E.: A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology* **15** (1982) 267–273
11. Cook, P.A.: *Nonlinear Dynamical Systems.* Prentice Hall, Englewood Cliffs, NJ (1986)