Sparse Coding Neural Gas: Learning of Overcomplete Data Representations

Kai Labusch, Erhardt Barth and Thomas Martinetz

University of Lübeck - Institute for Neuro- and Bioinformatics
Ratzeburger Allee 160 – 23538 Lübeck – Germany

Abstract

We consider the problem of learning an unknown (overcomplete) basis from data that are generated from unknown and sparse linear combinations. Introducing the Sparse Coding Neural Gas algorithm, we show how to employ a combination of the original Neural Gas algorithm and Oja's rule in order to learn a simple sparse code that represents each training sample by only one scaled basis vector. We generalize this algorithm by using Orthogonal Matching Pursuit in order to learn a sparse code where each training sample is represented by a linear combination of up to k basis elements. We evaluate the influence of additive noise and the coherence of the original basis on the performance with respect to the reconstruction of the original basis and compare the new method to other state of the art methods. For this analysis, we use artificial data where the original basis is known. Furthermore, we employ our method to learn an overcomplete representation for natural images and obtain an appealing set of basis functions that resemble the receptive fields of neurons in the primary visual cortex. An important result is that the algorithm converges even with a high degree of overcompleteness. A reference implementation of the methods is provided[1].

Key words: Sparse Coding, Vector Quantization, Matching Pursuit, Unsupervised Learning

1 Introduction

In the last few years there has been an increased interest in sparse coding. On the one hand, sparse coding is closely connected to independent component analysis (ICA), in particular to overcomplete and noisy ICA [20, 13, 36]. On the other hand, there is evidence that sparse coding is a principle employed

[1] http://www.inb.uni-luebeck.de/tools-demos/scng

*Article published in Neurocomputing **72**, 2009, pp. 1547–1555*
doi:10.1016/j.neucom.2008.11.027

by biological systems for signal processing [27, 29]; sparse models have been successfully used to mimic properties of simple cells in the primary visual cortex [7, 28, 2, 34]. More recent research has studied overcomplete sparse codes [22, 1, 19]. It has been shown that sparse codes possess favorable properties with respect to noise resistance at the reconstruction [37, 38] and representation level [5]. Applications for sparse coding range from compression [21] over denoising [9] to feature extraction [25, 33, 17].

Mathematically, the problem of sparse coding is to estimate a possibly overcomplete basis from given training samples $X = (\mathbf{x}_1, \ldots, \mathbf{x}_L)$, $\mathbf{x}_i \in \mathbb{R}^N$ that have been generated from an unknown sparse linear combination. Without loss of generality, we require X to have zero mean. We measure the quality of the basis by the mean square of the representation error:

$$E = \frac{1}{L} \sum_{i=1}^{L} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 \,. \tag{1}$$

$C = (\mathbf{c}_1, \ldots, \mathbf{c}_M)$, $\mathbf{c}_j \in \mathbb{R}^N$ denotes a matrix containing the basis elements. $\mathbf{a}_i \in \mathbb{R}^M$ denotes a set of sparse coefficients that have been chosen optimally for given $\mathbf{x}_i$ and C. The number of basis elements M is a free model parameter. In the case of overcomplete bases, $M > N$ holds. By imposing different constraints on the basis C or the choice of the coefficients $\mathbf{a}_i$, the structure of the learned basis can be controlled.

A number of methods that have been proposed consider a probabilistic setting that leads to the minimization of (1) under sparseness constraints with respect to the coefficients $\mathbf{a}_i$. These methods propose a data model where each given sample $\mathbf{x}_i$ is generated according to the following probabilistic model:

$$\mathbf{x}_i = C\mathbf{a} + \boldsymbol{\epsilon} \,. \tag{2}$$

The $\mathbf{a}$ are hidden variables that are sparsely distributed according to $P(\mathbf{a})$, which may for instance be a Laplacian. The residual $\boldsymbol{\epsilon}$ is assumed to be Gaussian. One wants to determine those model parameters C that maximize the probability of obtaining the observed data, i.e., that maximize the data likelihood:

$$\max_C P(\mathbf{x}_1, \ldots, \mathbf{x}_L | C) \tag{3}$$

with

$$P(\mathbf{x}_1, \ldots, \mathbf{x}_L | C) = \prod_{i=1}^{L} \int_{\mathbf{a}} P(\mathbf{x}_i | \mathbf{a}, C) P(\mathbf{a}) \mathrm{d}\mathbf{a} \,. \tag{4}$$

The well-known Sparsenet algorithm of Olshausen and Field [28, 29] and the algorithm proposed by Lewicki and Sejnowski [22] belong to the group of methods that consider such a probabilistic setting. These two algorithms differ in the way they deal with the intractable integration over $\mathbf{a}$ in (4). The related approach proposed by Kreutz-Delgado et al. [14] jointly maximizes the posterior probabilities of the model parameters C and $\mathbf{a}$:

$$\max_{C,\mathbf{a}_1,\ldots,\mathbf{a}_L} P(C,\mathbf{a}_1,\ldots,\mathbf{a}_L|\mathbf{x}_1,\ldots,\mathbf{x}_L) \tag{5}$$

which is equivalent to

$$\max_{C,\mathbf{a}_1,\ldots,\mathbf{a}_L} P(\mathbf{x}_1,\ldots,\mathbf{x}_L|C,\mathbf{a}_1,\ldots,\mathbf{a}_L)P(\mathbf{a}_1,\ldots,\mathbf{a}_L)P(C)\ . \tag{6}$$

The problematic integration with respect to $\mathbf{a}$ in (4) is avoided. Furthermore, the introduction of a prior with respect to C allows to directly incorporate constraints on the basis into the probabilistic setting.

ICA considers a setting that is very similar to (2). Assuming that the hidden variables $\mathbf{a} = (a_1, ..., a_M)$ are statistically independent, ICA looks for model parameters C that maximize the statistical independence of these hidden variables [13]. Statistical independence can be measured by non-Gaussianity, and maximization of non-Gaussianity leads to the sparsification of the hidden variables $\mathbf{a}$ in some cases [13]. Standard ICA considers the noise-free case where $\|\boldsymbol{\epsilon}\| = 0$ [13], but ICA methods that allow for additive noise have also been proposed [10]. Overcomplete ICA allows more underlying independent components C, i.e., hidden variables $\mathbf{a}$, than observed variables $\mathbf{x}_i$ [11, 20, 36, 3].

Given a fixed basis C, methods such as Matching Pursuit [35], Orthogonal Matching Pursuit [30] and Optimized Orthogonal Matching Pursuit [31] can be used to approximate those coefficients $\mathbf{a}_i$ that minimize (1) restricted to a maximum number of non-zero entries of the $\mathbf{a}_i$, i.e., these methods provide an approximation to the solution of the following optimization problem:

$$\min_{\mathbf{a}} \|\mathbf{x}_i - C\mathbf{a}\| \quad \text{subject to} \quad \|\mathbf{a}\|_0 \leq k\ . \tag{7}$$

Here, $\|\mathbf{a}\|_0$ denotes the number of non-zero coefficients in $\mathbf{a}$. Methods such as the MOD algorithm [6] and the K-SVD algorithm [1] have been proposed to learn the basis C by employing these pursuit methods. The pursuit method is used to determine $\mathbf{a}_1,\ldots,\mathbf{a}_L$. Then, in order to update C with respect to (1), $\mathbf{a}_1,\ldots,\mathbf{a}_L$ are considered fixed.

In this paper, we wish to show that sparse coding is also closely connected to the large field of vector quantization methods. We show how to learn an overcomplete sparse code under the presence of noise using an algorithm that is derived from the Neural Gas (NG) method [24, 23]. This method, the Sparse Coding Neural Gas algorithm, was first introduced in [16] and has already been applied to nuclear magnetic resonance data [32] and to the problem of blind separation of noisy overcomplete sources [18].

2 Sparse Coding Neural Gas

Let us start by considering a simple well-known approach for data representation: vector quantization. Vector quantization is based on a set of so-called codebook vectors. Each sample is encoded by the closest codebook vector. Therefore, for the coefficients $\mathbf{a}_i$, we have

$$a_{i_k} = 1,\ a_{i_j} = 0\ \forall j \neq k \quad \text{where} \quad k = \arg\min_j \|\mathbf{c}_j - \mathbf{x}_i\|_2^2\ . \tag{8}$$

Vector quantization aims to find a set of codebook vectors that minimize (1) under the constraints posed by (8). The well-known k-means algorithm [8] is one of the methods that try to solve this optimization problem. But k-means can lead to a sub-optimal utilization of the codebook vectors with respect to (1), i.e., bad quantization, due to the hard-competitive nature of its learning scheme. Furthermore, the k-means algorithm is initialization-sensitive and exhibits slow convergence. The Neural Gas algorithm [23] remedies these deficiencies by using a soft-competitive learning scheme that facilitates robust convergence to close to optimal distributions of the codebook vectors over the data manifold to be learned.

Here, we do not want to perform vector quantization but make a step towards a more flexible coding scheme, i.e., a coding scheme that in some cases may better resemble the structure of the data. We drop one constraint on the coefficients $\mathbf{a}_i$ to allow a representation in terms of an arbitrarily scaled single codebook vector. In other words, we are now looking for a set of one-dimensional subspaces that cover the data. This can be understood as considering a set of data directions instead of data modes. Due to the added flexibility of the coefficients, we require $\|\mathbf{c}_j\|_2^2 = 1$ without loss of generality. This leads to the following optimization problem, which can be understood as a model of maximum sparseness:

$$\min_{\mathbf{c}_1,\ldots,\mathbf{c}_M} \sum_{i=1}^{L} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 \quad \text{subject to} \quad \|\mathbf{a}_i\|_0 \leq 1 \quad \text{and} \quad \|\mathbf{c}_j\|_2^2 = 1\ . \tag{9}$$

First consider the marginal case of (9), where only one codebook vector is available, i.e, $M = 1$. Now (9) becomes:

$$\min_{\mathbf{c}} \sum_{i=1}^{L} \|\mathbf{x}_i - \mathbf{c}a_i\|_2^2 = \sum_{i=1}^{L} \mathbf{x}_i^T\mathbf{x}_i - 2a_i\mathbf{c}^T\mathbf{x}_i + a_i^2 \text{ subject to } \|\mathbf{c}\|_2^2 = 1\ . \tag{10}$$

When $\mathbf{x}_i$ and $\mathbf{c}$ are fixed, (10) becomes minimal by choosing $a_i = \mathbf{c}^T\mathbf{x}_i$. As the final optimization problem, one obtains:

$$\max_{\mathbf{c}} \sum_{i=1}^{L} (\mathbf{c}^T\mathbf{x}_i)^2 \quad \text{subject to} \quad \|\mathbf{c}\|_2^2 = 1\ . \tag{11}$$

Hence, in this marginal case, the problem of finding the codebook vector that minimizes (10) boils down to finding the direction of maximum variance. A well-known learning rule that solves (11), i.e., that finds the direction of maximum variance, is called Oja's rule [26]:

$$\Delta \mathbf{c} = \alpha \, y \, (\mathbf{x} - y \, \mathbf{c}) \tag{12}$$

with $y = \mathbf{c}^T \mathbf{x}$ and learning rate α.

Now consider the general case, where $M > 1$ holds. In this case, the optimization problem (11) turns into

$$\max_{\mathbf{c}_1,\ldots,\mathbf{c}_M} \sum_{i=1}^{L} \max_l (\mathbf{c}_l^T \mathbf{x}_i)^2 \quad \text{subject to} \quad \|\mathbf{c}_l\|_2^2 = 1 \; . \tag{13}$$

We can generalize to this case by first determining the codebook vector that has maximum overlap with respect to the training data:

$$l_{\text{win}} = \arg\max_l (\mathbf{c}_l^T \mathbf{x})^2 \; . \tag{14}$$

In order to minimize (9), we then update this codebook vector $\mathbf{c}_{l_{\text{win}}}$ according to Oja's rule. However, this approach suffers from the same problem as the k-means algorithm. Due to hard-competitive selection of the codebook vector to be updated, it may happen that the codebook vectors will be distributed sub-optimally with respect to the target function (see also Figure 1 in the experiments section). To prevent this, we modify the original Neural Gas algorithm (see Algorithm 1) to solve the general case of (9).

In the Neural Gas algorithm, soft-competitive learning is achieved by controlling the update of each codebook vector by its rank in the sequence of distances of all codebook vectors with respect to a given sample. These distances are computed within the sample space (see Algorithm 1, steps 4 and 5). We replace the distance measure and now consider the following sequence of distances (see Algorithm 2, step 4):

$$-\left(\mathbf{c}_{l_0}^T \mathbf{x}\right)^2 \leq \cdots \leq -\left(\mathbf{c}_{l_k}^T \mathbf{x}\right)^2 \leq \cdots \leq -\left(\mathbf{c}_{l_{M-1}}^T \mathbf{x}\right)^2 \; . \tag{15}$$

The modified distance measure requires a new update rule to minimize the distances between the codebook vectors and the current training sample $\mathbf{x}$. By combining Oja's rule with the soft-competitive update of the NG algorithm, we obtain (see Algorithm 2, step 5):

$$\Delta \mathbf{c}_{l_k} = \alpha_t e^{-k/\lambda_t} y \left(\mathbf{x} - y \mathbf{c}_{l_k}\right) \; . \tag{16}$$

Here, α_t is the learning rate, and λ_t is the neighbourhood size at time t:

$$\alpha_t = \alpha_0 \left(\alpha_{\text{final}} / \alpha_0\right)^{t/t_{\max}} \; , \tag{17}$$

$$\lambda_t = \lambda_0 \left(\lambda_{\text{final}}/\lambda_0\right)^{t/t_{\max}} \ . \tag{18}$$

For $t \to t_{\max}$, one obtains equation (12) as the update rule. Because of the optimization constraint $\|\mathbf{c}_j\| = 1$, we normalize the codebook vectors in each learning step. The complete Sparse Coding Neural Gas algorithm is shown in Algorithm 2.

3 On the convergence of Sparse Coding Neural Gas

Consider the following maximization problem:

$$\max_{\mathbf{c}_1,\ldots,\mathbf{c}_M} \sum_{i=1}^{L} \sum_{l=1}^{M} h_{\lambda_t}(k(\mathbf{c}_l, \mathbf{x}_i))(\mathbf{c}_l^T \mathbf{x}_i)^2 \quad \text{subject to} \quad \|\mathbf{c}_l\|_2^2 = 1, \tag{19}$$

with $h_{\lambda_t}(v) = e^{-v/\lambda_t}$. Let $k(\mathbf{c}_l, \mathbf{x})$ denote the number of basis elements $\mathbf{c}_j$ with $(\mathbf{c}_j^T \mathbf{x})^2 > (\mathbf{c}_l^T \mathbf{x})^2$. Note that for $\lambda_t \to 0$ this optimization problem is equivalent to the optimization problem defined by (13). In order to maximize (19), we consider the Lagrangian

$$L = \sum_{i=1}^{L} \sum_{l=1}^{M} h_{\lambda_t}(k(\mathbf{c}_l, \mathbf{x}_i))(\mathbf{c}_l^T \mathbf{x}_i)^2 - \beta_l(\mathbf{c}_l^2 - 1), \tag{20}$$

where we have introduced the Lagrangian multipliers β_l. We obtain

$$\frac{\partial L}{\partial \mathbf{c}_j} = 2 \sum_{i=1}^{L} h_{\lambda_t}(k(\mathbf{c}_j, \mathbf{x}_i))(\mathbf{c}_j^T \mathbf{x}_i)\mathbf{x}_i - 2\beta_j \mathbf{c}_j + R_j \tag{21}$$

with

$$R_j = \sum_{i=1}^{L} \sum_{l=1}^{M} h'_{\lambda_t}(k(\mathbf{c}_l, \mathbf{x}_i)) \frac{\partial k(\mathbf{c}_l, \mathbf{x}_i)}{\partial \mathbf{c}_j} (\mathbf{c}_l^T \mathbf{x}_i)^2 \tag{22}$$

and $h'_{\lambda_t}(v) = \frac{\partial h_{\lambda_t}(v)}{\partial v}$. Due to the arguments presented in [23], $R_j = 0$ holds.

At the maximum we have

$$\frac{\partial L}{\partial \mathbf{c}_j} = 0 \Leftrightarrow \beta_j = \sum_{i=1}^{L} h_{\lambda_t}(k(\mathbf{c}_j, \mathbf{x}_i))(\mathbf{c}_j^T \mathbf{x}_i)^2 \ . \tag{23}$$

Using this, we finally obtain the gradient

$$\frac{\partial L}{\partial \mathbf{c}_j} = 2 \sum_{i=1}^{L} h_{\lambda_t}(k(\mathbf{c}_j, \mathbf{x}_i))(\mathbf{c}_j^T \mathbf{x}_i)\mathbf{x}_i - (\mathbf{c}_j^T \mathbf{x}_i)^2 \mathbf{c}_j \ . \tag{24}$$

Hence, for a randomly chosen $\mathbf{x} \in (\mathbf{x}_1, \ldots, \mathbf{x}_L)$ at time t with learning rate α_t, the update

$$\begin{aligned}\Delta \mathbf{c}_j &= \alpha_t h_{\lambda_t}(k(\mathbf{c}_j,\mathbf{x}))\left((\mathbf{c}_j^T\mathbf{x})\mathbf{x} - (\mathbf{c}_j^T\mathbf{x})^2\mathbf{c}_j\right) && (25)\\ &= \alpha_t e^{-k(\mathbf{c}_j,\mathbf{x})/\lambda_t} y\,(\mathbf{x} - y\mathbf{c}_j) && (26)\end{aligned}$$

performs a stochastic gradient descent with respect to (19) [15]. Note that multiplying a basis element $\mathbf{c}_l$ by -1 does not change (19), therefore the sign of the basis elements cannot be recovered by minimizing (19).

Algorithm 1 The Neural Gas algorithm

```
1 initialize $C = (\mathbf{c}_1, \ldots, \mathbf{c}_M)$ using uniform random values
  for $t = 1$ to $t_{\max}$ do
2   select random sample $\mathbf{x}$ out of $X$
3   calculate current size of neighbourhood and learning rate:
        $\lambda_t = \lambda_0 \left(\lambda_{\text{final}}/\lambda_0\right)^{t/t_{\max}}$
        $\alpha_t = \alpha_0 \left(\alpha_{\text{final}}/\alpha_0\right)^{t/t_{\max}}$
4   determine the sequence $l_0, \ldots, l_{M-1}$ with:
        $\|\mathbf{x} - \mathbf{c}_{l_0}\| \leq \cdots \leq \|\mathbf{x} - \mathbf{c}_{l_k}\| \leq \cdots \leq \|\mathbf{x} - \mathbf{c}_{l_{M-1}}\|$
    for $k = 0$ to $M - 1$ do
5     update $\mathbf{c}_{l_k}$ according to $\mathbf{c}_{l_k} = \mathbf{c}_{l_k} + \alpha_t e^{-k/\lambda_t} \left(\mathbf{x} - \mathbf{c}_{l_k}\right)$
    end for
  end for
```

Algorithm 2 The Sparse Coding Neural Gas algorithm.

```
1 initialize $C = (\mathbf{c}_1, \ldots, \mathbf{c}_M)$ using uniform random values
  for $t = 1$ to $t_{\max}$ do
2   select random sample $\mathbf{x}$ out of $X$
3   set $\mathbf{c}_1, \ldots, \mathbf{c}_M$ to unit length
4   calculate current size of neighbourhood and learning rate:
        $\lambda_t = \lambda_0 \left(\lambda_{\text{final}}/\lambda_0\right)^{t/t_{\max}}$
        $\alpha_t = \alpha_0 \left(\alpha_{\text{final}}/\alpha_0\right)^{t/t_{\max}}$
    determine $l_0, \ldots, l_{M-1}$ with:
        $-(\mathbf{c}_{l_0}^T\mathbf{x})^2 \leq \cdots \leq -(\mathbf{c}_{l_k}^T\mathbf{x})^2 \leq \cdots \leq -(\mathbf{c}_{l_{M-1}}^T\mathbf{x})^2$
    for $k = 0$ to $M - 1$ do
5     with $y = \mathbf{c}_{l_k}^T\mathbf{x}$, update $\mathbf{c}_{l_k}$ according to $\mathbf{c}_{l_k} = \mathbf{c}_{l_k} + \alpha_t e^{-k/\lambda_t} y(\mathbf{x} - y\mathbf{c}_{l_k})$
    end for
  end for
```

4 Generalized Sparse Coding Neural Gas

The Generalized Sparse Coding Neural Gas (GSCNG) algorithm uses a linear combination of k elements of C to represent a given sample $\mathbf{x}_i$. It considers the following optimization problem:

$$\min_{\mathbf{c}_1,\ldots,\mathbf{c}_M} \sum_{i=1}^{L} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 \text{ subject to } \|\mathbf{a}_i\|_0 \leq k \text{ and } \|\mathbf{c}_j\|_2^2 = 1 \ . \tag{27}$$

Even if the optimal basis C is known, we still have to solve the following optimization problem:

$$\mathbf{a}_i^{\text{opt}} = \arg\min_{\mathbf{a}_i} \|\mathbf{x}_i - C\mathbf{a}_i\| \text{ subject to } \|\mathbf{a}_i\|_0 \leq k, \ i = 1, \ldots, L \ . \tag{28}$$

Here, $\mathbf{x}_i^{\text{opt}} = C\mathbf{a}_i^{\text{opt}}$ is the best k-term representation of $\mathbf{x}_i$ in terms of the given basis C.

In general, (28) is a combinatorial problem that is NP-hard [4]. A number of approximation methods have been proposed that tackle the problem of finding optimal coefficients $\mathbf{a}_i$ constrained by $\|\mathbf{a}_i\|_0 \leq k$ given fixed C and $\mathbf{x}_i$. We here consider a class of greedy methods, the so-called pursuit algorithms, that iteratively construct the vector $\mathbf{x}_i$ out of the columns of the matrix C.

4.1 Matching Pursuit (MP)

We start with a simple approach and consider the Matching Pursuit algorithm (MP). Let $C\mathbf{a}_i^{\text{MP}}$ denote the current approximation of $\mathbf{x}_i$ in MP, and let $\boldsymbol{\epsilon}_i = \mathbf{x}_i - C\mathbf{a}_i^{\text{MP}}$ denote the current residual that still has to be encoded. Initially, $\mathbf{a}_i^{\text{MP}} = 0$ and $\boldsymbol{\epsilon}_i = \mathbf{x}_i$. MP iteratively selects k columns of C by performing the following steps:

(1) Select $\mathbf{c}_{l_{\text{win}}}$ by $\mathbf{c}_{l_{\text{win}}} = \arg\max_{\mathbf{c}_l}(\mathbf{c}_l^T\boldsymbol{\epsilon}_i)$
(2) Set $(\mathbf{a}_i^{\text{MP}})_{l_{\text{win}}} = (\mathbf{a}_i^{\text{MP}})_{l_{\text{win}}} + (\mathbf{c}_{l_{\text{win}}}^T\boldsymbol{\epsilon}_i)$
(3) Obtain new residual $\boldsymbol{\epsilon}_i = \mathbf{x}_i - C\mathbf{a}_i^{\text{MP}}$
(4) Continue with step 1 until k iterations have been performed

Even if we perform N iterations of MP, i.e., if we select as many basis vectors as there are dimensions, it is not guaranteed that we will obtain $C\mathbf{a}_i^{\text{MP}} = \mathbf{x}_i$ and $\boldsymbol{\epsilon}_i = 0$, though the asymptotical convergence of MP for $k \rightarrow \infty$ has been proven [35].

4.2 Orthogonal Matching Pursuit (OMP)

Let $C\mathbf{a}_i^{\mathrm{OMP}}$ denote the current approximation of $\mathbf{x}_i$ in Orthogonal Matching Pursuit. In contrast to MP, this approximation fulfills $C\mathbf{a}_i^{\mathrm{OMP}} = \mathbf{x}_i$ and $\boldsymbol{\epsilon}_i = 0$ after $k \leq N$ iterations [30]. Let U denote the set of indices of those columns of C that already have been used during Orthogonal Matching Pursuit. The number of elements in U, i.e., $|U|$, equals the number of iterations that have been performed so far. The columns of C that are indexed by U are denoted by C^U. Initially, $\mathbf{a}_i^{\mathrm{OMP}} = 0$, $\boldsymbol{\epsilon}_i = \mathbf{x}_i$ and $U = \emptyset$. OMP works as follows:

(1) Select $\mathbf{c}_{l_{\mathrm{win}}}$ by $\mathbf{c}_{l_{\mathrm{win}}} = \arg\max_{\mathbf{c}_l, l \notin U}(\mathbf{c}_l^T \boldsymbol{\epsilon}_i)$
(2) Set $U = U \cup l_{\mathrm{win}}$
(3) Solve the optimization problem $\mathbf{a}_i^{\mathrm{OMP}} = \arg\min_{\mathbf{a}} \|\mathbf{x}_i - C^U \mathbf{a}\|_2^2$
(4) Obtain current residual $\boldsymbol{\epsilon}_i = \mathbf{x}_i - C\mathbf{a}_i^{\mathrm{OMP}}$
(5) Continue with step 1 until k iterations have been performed

An important property of the basis C that has an impact on the quality of the approximation provided by the OMP algorithm is the mutual coherence $H(C)$ of the basis C:

$$H(C) = \max_{1 \leq i,j \leq M, i \neq j} |\mathbf{c}_i^T \mathbf{c}_j| \ . \tag{29}$$

It has been shown that OMP yields an $\mathbf{x}_i^{\mathrm{OMP}} = C\mathbf{a}_i^{\mathrm{OMP}}$ with

$$\|\mathbf{x}_i - \mathbf{x}_i^{\mathrm{OMP}}\| \leq \sqrt{1+6k}\ \|\mathbf{x}_i - \mathbf{x}_i^{\mathrm{opt}}\| \tag{30}$$

if the mutual coherence of C is small enough [37].

4.3 Optimized Orthogonal Matching Pursuit (OOMP)

An improved variant of the OMP algorithm is Optimized Orthogonal Matching Pursuit (OOMP) [31]. In general, the columns of C are not pairwise orthogonal. Hence, the criterion of OMP that selects the column $\mathbf{c}_{l_{\mathrm{win}}}, l_{\mathrm{win}} \notin U$ of C that is added to U is not optimal with respect to the minimization of the residual that is obtained after the column $\mathbf{c}_{l_{\mathrm{win}}}$ has been added. Therefore, Optimized Orthogonal Matching Pursuit uses a selection criterion that is optimal with respect to the minimization of the norm of the residual obtained: the algorithms runs through all columns of C that have not been used so far and selects the one that yields the smallest residual. Optimized Orthogonal Matching Pursuit works as follows:

(1) Select $\mathbf{c}_{l_{\mathrm{win}}}$ such that $\mathbf{c}_{l_{\mathrm{win}}} = \arg\min_{\mathbf{c}_l, l \notin U} \min_{\mathbf{a}} \|\mathbf{x} - C^{U \cup l}\mathbf{a}\|$

(2) Set $U = U \cup l_{\text{win}}$

(3) Solve the optimization problem $\mathbf{a}_i^{\text{OMP}} = \arg\min_{\mathbf{a}} \|\mathbf{x}_i - C^U\mathbf{a}\|_2^2$

(4) Obtain current residual $\boldsymbol{\epsilon}_i = \mathbf{x}_i - C\mathbf{a}_i^{\text{OMP}}$

(5) Continue with step 1 until k iterations have been performed

The selection criterion of the OOMP algorithm (step 1) involves $M - |U|$ minimization problems, one for each column of C that has not been used so far. In order to reduce the computational complexity of this step, we use an implementation of the OOMP algorithm that employs a temporary matrix R that has been orthogonalized with respect to C^U. R is obtained by removing the projection of the columns of C onto the subspace spanned by C^U from C and setting the norm of the residuals $\mathbf{r}_l$ to one. The residual $\boldsymbol{\epsilon}_i^U$ is obtained in the same way, i.e., the projection of $\mathbf{x}_i$ to the subspace spanned by C^U is removed from $\mathbf{x}_i$. Initially, $R = (\mathbf{r}_1, \ldots, \mathbf{r}_l, \ldots, \mathbf{r}_M) = C$ and $\boldsymbol{\epsilon}_i^U = \mathbf{x}_i$. In each iteration, the algorithm determines the column $\mathbf{r}_l$ of R with $l \notin U$ that has maximum overlap with respect to the current residual $\boldsymbol{\epsilon}_i^U$:

$$l_{\text{win}} = \arg\max_{l, l \notin U} (\mathbf{r}_l^T \boldsymbol{\epsilon}_i^U)^2 \ . \tag{31}$$

Then, in the construction step, the orthogonal projection with respect to $\mathbf{r}_{l_{\text{win}}}$ is removed from the columns of R and $\boldsymbol{\epsilon}_i^U$:

$$\mathbf{r}_l = \mathbf{r}_l - (\mathbf{r}_{l_{\text{win}}}^T \mathbf{r}_l)\mathbf{r}_{l_{\text{win}}}, \tag{32}$$

$$\boldsymbol{\epsilon}_i^U = \boldsymbol{\epsilon}_i^U - (\mathbf{r}_{l_{\text{win}}}^T \boldsymbol{\epsilon}_i^U)\mathbf{r}_{l_{\text{win}}} \ . \tag{33}$$

After the projection has been removed, l_{win} is added to U, i.e., $U = U \cup l_{\text{win}}$. The columns $\mathbf{r}_l$ with $l \notin U$ may be selected in the subsequent iterations of the algorithm. The norm of these columns is set to unit length. If the stopping criterion $|U| = k$ has been reached, the final entries of $\mathbf{a}_i^{\text{OMP}}$ can be obtained by recursively collecting the contribution of each column of C during the construction process, taking into account the normalization of the columns of R in each iteration.

4.4 Learning the basis C

So far, we have considered the case where the optimal basis C is given. The Generalized Sparse Coding Neural Gas algorithm shall learn the optimal basis, i.e., we now consider the problem of learning the basis $C = (\mathbf{c}_1, \ldots, \mathbf{c}_M)$ from the training data $\mathbf{x}_i$ provided that we know the number of basis elements M and the dimension k of the subspaces that cover the training data. We use OOMP to realize this generalization of SCNG: In each iteration of GSCNG, the OOMP algorithm is performed. In order to minimize (27), we perform an update of R and C prior to the construction step (32) and (33) in each of the

k iterations of OOMP. The update step reduces the norm of the residual that is obtained in the current iteration. The norm of the residual becomes small if

$$(\mathbf{r}_{l_{\mathrm{win}}}^T \boldsymbol{\epsilon}_i^U)^2 \tag{34}$$

is large. Hence, we have to consider the optimization problem

$$\max_{\mathbf{r}_1,\ldots,\mathbf{r}_{M-|U|}} \sum_{i=1}^{L} \max_{l,l\notin U} (\mathbf{r}_l^T \boldsymbol{\epsilon}_i^U)^2 \quad \text{subject to} \quad \|\mathbf{r}_l\| = 1 \ . \tag{35}$$

The optimization problem (35) is very similar to (13), but now we consider the data $\boldsymbol{\epsilon}_i^U$ as well as the codebook vectors $\mathbf{r}_l$ that have been orthogonalized with respect to those codebook vectors C^U that have already been used during OOMP. As before, an optimization of (35) can be achieved by using Oja's rule. Instead of updating only the winning column of R, i.e, $\mathbf{r}_{l_{\mathrm{win}}}$, we again employ the soft-competitive learning approach of the NG algorithm in order to update each column of R that may be selected in the next iteration of the OOMP algorithm. Again, we determine a sequence of distances of the current training sample to the current codebook vectors. But now, we only consider distances in the subspace that is orthogonal to C^U (see Algorithm 3, step 3):

$$-\left(\mathbf{r}_{l_0}^T \boldsymbol{\epsilon}_i^U\right)^2 \leq \cdots \leq -\left(\mathbf{r}_{l_k}^T \boldsymbol{\epsilon}_i^U\right)^2 \leq \cdots \leq -\left(\mathbf{r}_{l_{M-|U|-1}}^T \boldsymbol{\epsilon}_i^U\right)^2 \quad , \ l_k \notin U \ . \tag{36}$$

As before, we combine Oja's rule and the soft-competitive update of the NG algorithm, but the update is now orthogonal to the subspace spanned by C^U. On the one hand, we apply the update to the temporary basis R; on the other hand, we accumulate the updates of all subsequent OOMP iterations in the learned mixing matrix C. Due to the orthogonal projection (32) and (33) performed in each iteration, these updates are pairwise orthogonal (see Algorithm 3, step 4):

$$\Delta\mathbf{r}_{l_k} = \Delta\mathbf{c}_{l_k} = \alpha_t e^{-k/\lambda_t} y \left(\boldsymbol{\epsilon}_i^U - y\, \mathbf{r}_{l_k}\right) \ . \tag{37}$$

This update rule corresponds to a stochastic gradient descent with respect to (35) because the arguments provided in Section 3 can be applied in the same way.

4.5 Computational time complexity

We do not provide an in-depth analysis of the time complexity of the algorithm but give a rough estimation.

Each update step can be split into the following tasks:

- $M-|U|$ distances with respect to the current residual have to be computed. The time complexity of this operation is $\mathcal{O}(MN)$.
- The distances have to be sorted and the winning basis vector has to be determined. This can be accomplished with a time complexity of $\mathcal{O}(M\log(M))$.
- The winning basis vector as well as those $M-|U|$ basis vectors that may be used in the subsequent steps have to be updated using the learning rule. This can be performed in $\mathcal{O}(MN)$ operations.
- The residual and $M-|U|$ remaining basis vectors of size N have to be orthogonalized with respect to the winning basis vector. The time complexity of this operation is $\mathcal{O}(MN)$.

Therefore, each update step has a computational time complexity of $\mathcal{O}(MN+M\log(M))$. Each iteration of the Generalized Sparse Coding Neural Gas algorithm performs k update steps, i.e., each iteration has a computational time complexity of $\mathcal{O}(k(MN+M\log(M)))$. Overall, $t_{\max}$ iterations are performed, therefore the overall time complexity of the algorithm is $\mathcal{O}(t_{\max}k(MN+M\log(M)))$.

The entire Generalized Sparse Coding Neural Gas method is shown in Algorithm 3.

5 Experiments

First, we test the Sparse Coding Neural Gas algorithm on artificially generated sparse linear combinations. We do not consider the task of determining M and k, i.e., the size of the basis that was used to generate the samples and the number of non-zero coefficients in each linear combination; instead, we assume M and k to be known.

The basis vectors and coefficients used to generate training samples are chosen from a uniform distribution. In order to study the impact of the mutual coherence (29) of the basis on the reconstruction performance, we vary the mutual coherence of the basis. We obtain a random basis with coherence z by repeatedly choosing a matrix from a uniform distribution in $[-1,1]$ until $\lceil 100H(C)\rceil = \lceil 100z\rceil$. Then, the norm of the columns of the basis matrix is set to unit length. The mean variance of the training samples is set to 1. A certain amount of uniformly distributed noise is added to the training samples.

First, we consider a two-dimensional toy example, where each training sample is a multiple of one of five basis vectors, i.e., $M=5, k=1, N=2$. The variance of the additive noise is set to 0.01. Figure 1 shows the training samples, the original basis C^{orig} (dashed lines) and the basis C^{learn} that was learned from the data (solid lines). The left part of the figure shows the result obtained by

Algorithm 3 The Generalized Sparse Coding Neural Gas algorithm.

```
    initialize $C = (\mathbf{c}_1, \dots, \mathbf{c}_M)$ using uniform random values
    for $t = 1$ to $t_{\max}$ do
1     select random sample $\mathbf{x}$ out of $X$
2     set $\mathbf{c}_1, \dots, \mathbf{c}_M$ to unit length
      calculate current size of neighbourhood: $\lambda_t = \lambda_0 \left(\lambda_{\text{final}}/\lambda_0\right)^{t/t_{\max}}$
      calculate current learning rate: $\alpha_t = \alpha_0 \left(\alpha_{\text{final}}/\alpha_0\right)^{t/t_{\max}}$
      set $U = \emptyset$, $\boldsymbol{\epsilon}^U = \mathbf{x}$ and $R = (\mathbf{r}_1, \dots, \mathbf{r}_M) = C = (\mathbf{c}_1, \dots, \mathbf{c}_M)$
      for $h = 0$ to $K - 1$ do
3       determine $l_0, \dots, l_k, \dots, l_{M-h-1}$ with $l_k \notin U$ :
            $$-(\mathbf{r}_{l_0}^T \boldsymbol{\epsilon}^U)^2 \leq \dots \leq -(\mathbf{r}_{l_k}^T \boldsymbol{\epsilon}^U)^2 \leq \dots \leq -(\mathbf{r}_{l_{M-h-1}}^T \boldsymbol{\epsilon}^U)^2$$
        for $k = 0$ to $M - h - 1$ do
4         with $y = \mathbf{r}_{l_k}^T \boldsymbol{\epsilon}^U$, update $\mathbf{c}_{l_k} = \mathbf{c}_{l_k} + \Delta_{l_k}$ and $\mathbf{r}_{l_k} = \mathbf{r}_{l_k} + \Delta_{l_k}$ with
              $$\Delta_{l_k} = \alpha_t e^{-k/\lambda_t} y(\boldsymbol{\epsilon}^U - y\mathbf{r}_{l_k})$$
          set $\mathbf{r}_{l_k}$ to unit length
        end for
5       determine $l_{\text{win}} = \arg \max_{l \notin U} (\mathbf{r}_l^T \boldsymbol{\epsilon}^U)^2$
6       remove projection to $\mathbf{r}_{l_{\text{win}}}$ from $\boldsymbol{\epsilon}^U$ and $R$:
            $$\boldsymbol{\epsilon}^U = \boldsymbol{\epsilon}^U - (\mathbf{r}_{l_{\text{win}}}^T \boldsymbol{\epsilon}^U)\mathbf{r}_{l_{\text{win}}}$$
            $$\mathbf{r}_l = \mathbf{r}_l - (\mathbf{r}_{l_{\text{win}}}^T \mathbf{r}_l)\mathbf{r}_{l_{\text{win}}} \quad ,l = 1, \dots, M \wedge \ l \notin U$$
7       set $U = U \ \cup \ l_{\text{win}}$
      end for
    end for
```

hard-competitive learning, i.e., $\lambda_0 = \lambda_{\text{final}} = 0$. Note that some of the original basis vectors are not learned correctly due to the sub-optimal distribution of the learned basis with respect to the given training data. The right part shows the result obtained using soft-competitive learning, i.e., $\lambda_0 = 5/2, \lambda_{\text{final}} = 0.01$. Note that the original basis is obtained except for the sign of the basis vectors.

In a second experiment, a basis $C^{\text{orig}} \in \mathbb{R}^{40 \times 100}$ is generated, consisting of $M = 100$ basis vectors of dimension 40. Linear combinations $\mathbf{x}_1, \dots, \mathbf{x}_{10000}$ of k basis vectors are computed using uniformly distributed coefficients in $[-1, 1]$. We generate different bases with mutual coherence $H(C) = 0.3, 0.4, 0.5, 0.6$. The learned basis C^{learn} is compared to the original basis C^{orig} that was used to generate the samples. This is done by taking the maximum overlap of each original basis vector $\mathbf{c}_j^{\text{orig}}$ and the learned basis vectors, i.e., $\max_i \ |\mathbf{c}_i^{\text{learn}} \mathbf{c}_j^{\text{orig}}|$. To assess how many of the learned basis vectors can be assigned unambigu-

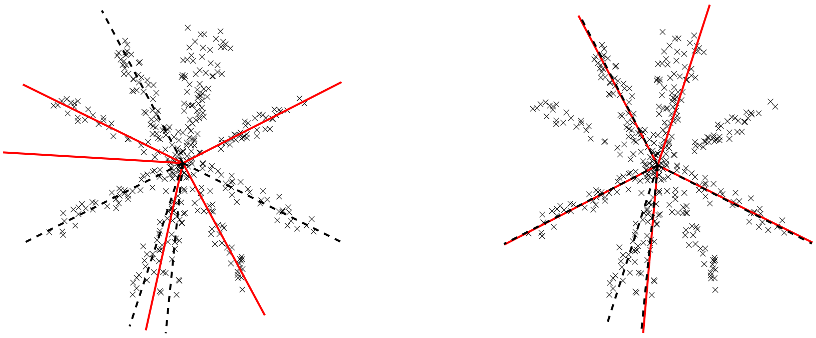

Fig. 1. A two-dimensional toy example where each sample is a multiple of one of five basis vectors plus additive noise. Left: hard-competitive learning, i.e., $\lambda_0 = \lambda_{\text{final}} = 0$. Some of the original basis vectors (dashed lines) are not learned correctly. Right: soft-competitive learning $\lambda_0 = 5/2, \lambda_{\text{final}} = 0.01$. The original basis is obtained except for the sign of the basis vectors. Note that though the data is radially arranged around the center of gravity in this toy example, this is not required for the method to work.

ously to the original basis, we consider $|\hat{C}^{\text{learn}}|$, which is the size of the set

$$\hat{C}^{\text{learn}} = \{\mathbf{c}_k^{\text{learn}} : k = \arg\max_i \ |\mathbf{c}_i^{\text{learn}}\mathbf{c}_j^{\text{orig}}|, \ j = 1, \ldots, M\} \ . \tag{38}$$

All experiments were repeated 10 times.

Figure 2 shows the impact of the mutual coherence of the basis on the mean maximum overlap and on the mean of $|\hat{C}^{\text{learn}}|$ for $k = 1, \ldots, 15$. It can be seen that the smaller the mutual coherence of the underlying basis is, the better the reconstruction performance. The amplitude of the additive noise was set to 0.1. Figure 3 shows the impact of the variance of the additive noise on the mean maximum overlap and on the mean of $|\hat{C}^{\text{learn}}|$. An increasing noise level leads to decreasing performance, as expected. Figure 2 and Figure 3 show that the less sparse the coefficients are (the larger k is), the lower the quality of the dictionary reconstruction (see also [37, 5]).

In the third experiment, we fix $k = 9$ and evaluate the reconstruction error (1) during the learning process while varying the noise amplitude and the mutual coherence of the basis. The coefficients used for reconstruction are determined by Optimized Orthogonal Matching Pursuit with k steps. Figure 4 shows that the reconstruction error decreases over time. The smaller the noise level is, the smaller the remaining reconstruction error. The mutual coherence of the basis has only slight influence on the remaining reconstruction error.

Finally, in order to compare the performance of the algorithm to other methods, we repeat the experiment described in [1]. A basis $C^{\text{orig}} \in \mathbb{R}^{20 \times 50}$ is generated, consisting of $M = 50$ basis vectors of dimension 20. Linear combinations $\mathbf{x}_1, \ldots, \mathbf{x}_{1500}$ of $k = 3$ basis vectors are computed using uniformly distributed coefficients. We add Gaussian noise to obtain data with varying SNR. We ob-

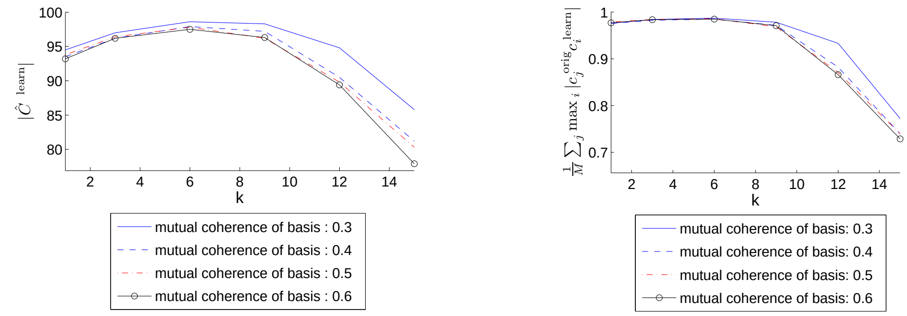

Fig. 2. The impact of the mutual coherence $H(C)$ on the performance of Sparse Coding Neural Gas. We used $M = 100$ basis vectors of dimension 40. Left: mean size of $\hat{C}^{\text{learn}}$. Right: mean maximum overlap between original and learned basis. The larger the mutual coherence of the basis is and the less sparse the linear combinations are, the more the performance decreases. Sparse Coding Neural Gas parameters used: $\lambda_0 = M/2, \lambda_{\text{final}} = 0.01, \alpha_0 = 0.1, \alpha_{\text{final}} = 0.0001, t_{\max} = 10 * 10000$. The noise variance was set to 0.1.

tain the learned basis by applying the Sparse Coding Neural Gas algorithm to the data. In [1], the number of learning iterations was set to 80, where each learning iteration uses the entire data. Therefore, we set $t_{\max} = 80 * 1500$. As in [1] we compare the learned basis to the original basis using the maximum overlap between each original basis vector and the learned basis, i.e, whenever

$$\max_j \left(1 - |\mathbf{c}_i^{\text{orig}} \mathbf{c}_j^{\text{learn}}|\right) \tag{39}$$

is smaller than 0.01, we count this as a success. We repeat this experiment 50 times with a varying SNR of 10dB, 20dB and 30dB as well as zero noise. As in [1], for each noise level we sort the 50 trials according to the number of successfully learned basis elements and order them in groups of ten experiments. Figure 5 shows the mean number of successfully detected basis vectors for each of the ten groups for each noise level. For comparison, the results for the MOD method [6], the method of Kreutz-Delgado (MAP) [14] and for the K-SVD method [1], taken from [1], are shown in Figure 5.

It can be seen that the Sparse Coding Neural Gas method outperforms the MAP method for all noise levels and performs as good as MOD for the 20dB and 30dB SNR and noise-free settings. Surprisingly, the performance of SCNG degrades at the 10dB SNR setting; this has to be investigated further. K-SVD outperforms Sparse Coding Neural Gas. It should be noted that K-SVD and MOD are batch methods that use the entire data in order to obtain the next update of the basis C in each learning iteration, whereas Sparse Coding Neural Gas is a pattern-by-pattern online method that only uses one data sample at a time. The development of a batch version of SCNG will be future work.

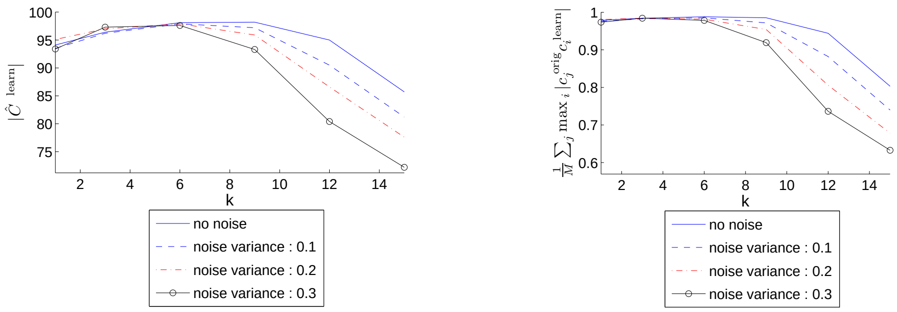

Fig. 3. The impact of the noise level on the performance of Sparse Coding Neural Gas. We used $M = 100$ basis vectors of dimension 40. Left: mean size of $\hat{C}^{\text{learn}}$. Right: mean maximum overlap between original and learned basis. The more noise is present and the less sparse the linear combinations are, the more the performance decreases. Sparse Coding Neural Gas parameters used: $\lambda_0 = M/2, \lambda_{\text{final}} = 0.01, \alpha_0 = 0.1, \alpha_{\text{final}} = 0.0001, t_{\max} = 10 * 10000$. The coherence of the basis was set to 0.4.

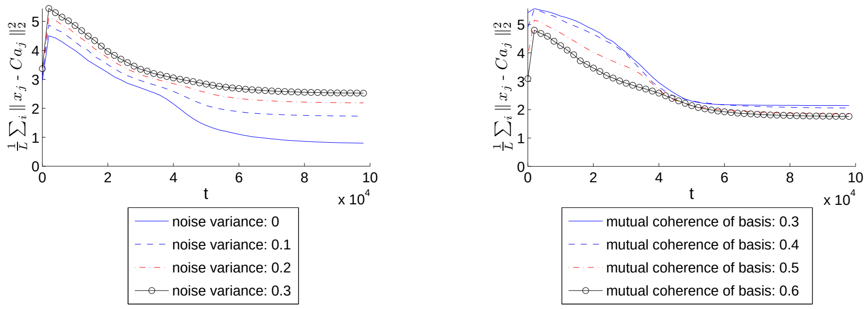

Fig. 4. Mean reconstruction error over time. We used $M = 100$ basis vectors of dimension 40 and set $k = 9$. Left: Impact of different noise levels on the reconstruction performance. The mutual coherence of the basis was set to 0.4. Right: Impact of the mutual coherence of the basis on the reconstruction performance. The noise variance was set to 0.1. The more noise is present, the larger the remaining reconstruction error. The mutual coherence has only slight influence on the remaining reconstruction error.

6 Experiments on natural image data

We used the SCNG algorithm to learn an overcomplete representation of random patches of natural images. The image patches of size 8×8 pixels were chosen randomly out of a number of landscape photographs published by Olshausen together with the Sparsenet algorithm. In order to reduce the influence of low frequencies on the reconstruction error, the images were bandpass filtered as described in [28]. The learned representation is 6.25 times overcomplete, i.e., it consists of 400 basis vectors of size $8 \times 8 = 64$. k, the number of non-zero entries per linear combination, was set to 30.

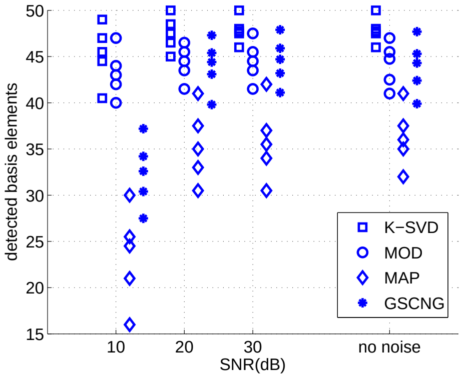

Fig. 5. Comparison of the performance of Generalized Sparse Coding Neural Gas (GSCNG) with respect to the reconstruction of the original basis on artificial data. The performance of MOD, K-SVD,MAP and GSCNG in the same setting are shown. The results for MOD, K-SVD and MAP were taken from [1]. GSCNG outperforms MAP and performs as good as MOD except on the 10dB SNR setting. K-SVD outperforms GSCNG.

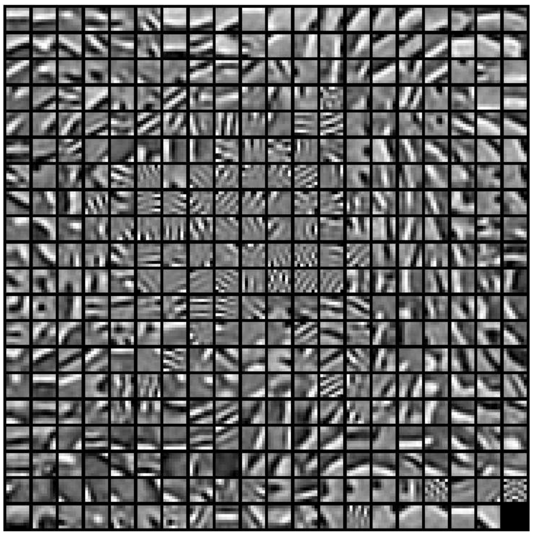

Fig. 6. A 6.25-times overcomplete basis of patches of natural images of size 8×8 pixels that was obtained by applying Sparse Coding Neural Gas to natural image data. The basis functions were arranged by mapping the basis vectors to a 2D grid using a Kohonen map.

Similar experiments have been performed by a number of researchers. They report the emergence of basis elements that, like Gabor wavelets, resemble properties of simple cells in the visual cortex, i.e., they obtain bandpass-like basis functions that are localized in space and orientiation [28, 2, 12]. An overcomplete basis of these patches of natural images obtained using the Sparse Coding Neural Gas algorithm is shown in Figure 6. It can be seen that the results reported by other researchers can be reproduced, i.e., we obtain bandpass-like structures ranging over different scales and localized in space and orientation.

7 Conclusion

We have described a new method, the Sparse Coding Neural Gas algorithm, to learn an overcomplete basis for the sparse encoding of a given data set. We used artificial data that was actually generated from a sparse linear combination of some original basis to assess the performance of our method with respect to the reconstruction of the original basis. We evaluated the influence of additive noise, and the mutual coherence of the original basis on reconstruction performance. Our experiments show that the performance obtained depends on the sparsity of the linear combinations, the strength of the additive noise and on the mutual coherence (degree of non-orthogonality) of the underlying basis. The sparser the linear combinations, the smaller the mutual coherence, and the lower the noise level the better the performance becomes. On an artificial data set that has been used by others as a performance measure, our method yields similar results to other state-of-the-art methods. Though it is an online method that learns pattern-by-pattern, it performs as well as state-of-the art batch methods which use the entire data in each learning iteration.

When applying the Sparse Coding Neural Gas algorithm to natural image data, we obtain bandpass-like basis elements localized in space and orientation. This reproduces results that have been reported by others for the sparse coding of natural images and shows that the Sparse Coding Neural Gas algorithm works robustly on real data. Moreover, the Kohonen mapping of the obtained "receptive fields" reveals, that a more natural sampling of the relevant parameter space is achieved. A further benefit of the algorithm is that it converges even in the case of highly overcomplete bases. Sparse coding generates favorable features for pattern recognition, as, e.g., demonstrated in [17]. We expect SCNG to be a very efficient method for constructing these sparse coding features. Another interesting application of the Sparse Coding Neural Gas method, the online learning of time-varying overcomplete sparse codes, will be pursued as future work, as well as the development of a batch version.

References

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, 54(11):4311–4322, 2006.

[2] A. J. Bell and T. J. Sejnowski. The "independent components" of natural scenes are edge filters. *Vision Res*, 37(23):3327–3338, December 1997.

[3] M. Davies and N. Mitianoudis. Simple mixture model for sparse overcomplete ICA. *IEEE Proceedings on Vision, Image and Signal Processing*, 151(1):35–43, 2004.

[4] G. Davis, S. Mallat, and M. Avellaneda. Greedy adaptive approximation. *J. Constr. Approx.*, 13:57–89, 1997.

[5] David L. Donoho, Michael Elad, and Vladimir N. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, 52(1):6–18, 2006.

[6] Kjersti Engan, Sven Ole Aase, and John Hrakon Husøy. Multi-frame compression: theory and design. *Signal Process.*, 80(10):2121–2140, 2000.

[7] David J. Field. What is the goal of sensory coding? *Neural Computation*, 6(4):559–601, 1994.

[8] J. A. Hartigan and M. A. Wong. A K-means Clustering Algorithm. *Applied Statistics*, 28:100–108, 1979.

[9] Patrik Hoyer and Erkki Oja. Image denoising by sparse code shrinkage. In *Intelligent Signal Processing*. IEEE Press, 2000.

[10] A. Hyvärinen. Gaussian moments for noisy independent component analysis. *IEEE Signal Processing Letters*, 6(6):145–147, 1999.

[11] A. Hyvarinen, R. Cristescu, and E. Oja. A fast algorithm for estimating overcomplete ICA bases for image windows. *Proceedings of the International Joint Conference on Neural Networks, IJCNN'99*, 2:894–899, 1999.

[12] Aapo Hyvärinen and Patrik Hoyer. Emergence of Phase- and Shift-Invariant Features by Decomposition of Natural Images into Independent Feature Subspaces. *Neural Comput.*, 12(7):1705–1720, 2000.

[13] Aapo Hyvarinen, Juha Karhunen, and Erkki Oja. *Independent Component Analysis*. Wiley-Interscience, May 2001.

[14] Kenneth Kreutz-Delgado, Joseph F. Murray, Bhaskar D. Rao, Kjersti Engan, Te-Won Lee, and Terrence J. Sejnowski. Dictionary learning algorithms for sparse representation. *Neural Comput.*, 15(2):349–396, 2003.

[15] H. J. Kushner and D. S. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Springer, 1978.

[16] Kai Labusch, Erhardt Barth, and Thomas Martinetz. Learning data representations with Sparse Coding Neural Gas. In Michel Verleysen, editor, *Proceedings of the 16th European Symposium on Artificial Neural Networks*, pages 233–238. D-Side Publishers, 2008.

[17] Kai Labusch, Erhardt Barth, and Thomas Martinetz. Simple Method for High-Performance Digit Recognition Based on Sparse Coding. *IEEE Transactions on Neural Networks*, 19(11):1985–1989, 2008.

[18] Kai Labusch, Erhardt Barth, and Thomas Martinetz. Sparse Coding Neural Gas for the Separation of Noisy Overcomplete Sources. In Vera Kurková, Roman Neruda, and Jan Koutník, editors, *Artificial Neural Networks - ICANN 2008, 18th International Conference, Prague, Czech Republic, September 3-6, 2008, Proceedings, Part II*, volume 5163 of *Lecture Notes in Computer Science*, pages 788–797. Springer, 2008.

[19] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 801–808. MIT Press, Cambridge, MA, 2007.

[20] Te-Won Lee, M.S. Lewicki, M. Girolami, and T.J. Sejnowski. Blind source separation of more sources than mixtures using overcomplete representations.

IEEE Signal Processing Letters, 6(4):87–90, 1999.

[21] Michael S. Lewicki and Bruno A. Olshausen. Probabilistic framework for the adaptation and comparison of image codes. *J. Opt. Soc. Am. A*, 16(7):1587–1601, 1999.

[22] Michael S. Lewicki and Terrence J. Sejnowski. Learning Overcomplete Representations. *Neural Computation*, 12(2):337–365, 2000.

[23] T. Martinetz, S. Berkovich, and K. Schulten. "Neural-gas" Network for Vector Quantization and its Application to Time-Series Prediction. *IEEE-Transactions on Neural Networks*, 4(4):558–569, 1993.

[24] T. Martinetz and K. Schulten. A "Neural-Gas Network" Learns Topologies. *Artificial Neural Networks*, I:397–402, 1991.

[25] Jim Mutch and David G. Lowe. Multiclass Object Recognition with Sparse, Localized Features. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 11–18, Washington, DC, USA, 2006. IEEE Computer Society.

[26] E. Oja. A simplified neuron model as a principal component analyzer. *J. Math. Biol.*, 15:267–273, 1982.

[27] B. Olshausen and D. Field. Sparse coding of natural images produces localized, oriented, bandpass receptive fields. Technical Report CCN-110-95, Department of Psychology, Cornell University, Ithaca, New York 14853, 1995.

[28] Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, (381):607–609, 1996.

[29] Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

[30] Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition. *Proceedings of the 27 th Annual Asilomar Conference on Signals, Systems,*, November 1993.

[31] L. Rebollo-Neira and D. Lowe. Optimized orthogonal matching pursuit approach. *IEEE Signal Processing Letters*, 9(4):137–140, 2002.

[32] Frank-Michael Schleif, Matthias Ongyerth, and Thomas Villmann. Sparse Coding Neural Gas for Analysis of Nuclear Magnetic Resonance Spectroscopy. In *CBMS*, pages 620–625. IEEE Computer Society, 2008.

[33] Thomas Serre, Lior Wolf, Stanley Bileschi, Maximilian Riesenhuber, and Tomaso Poggio. Robust Object Recognition with Cortex-Like Mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):411–426, 2007.

[34] Eero P. Simoncelli and Bruno A. Olshausen. Natural image statistics and neural representation. *Annual Review of Neuroscience*, 24:1193–1216, 2001.

[35] S.Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41:3397–3415, 1993.

[36] F. Theis, E. Lang, and C. Puntonet. A Geometric Algorithm for Overcomplete Linear ICA. *Neurocomputing*, 56:381–398, 2004.

[37] J. A. Tropp. Greed is good: algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.

[38] J. A. Tropp and A. C. Gilbert. Signal Recovery From Random Measurements

Via Orthogonal Matching Pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007.