Soft-competitive Learning of Sparse Codes and its Application to Image Reconstruction

Kai Labusch, Erhardt Barth and Thomas Martinetz

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

Abstract

We propose a new algorithm for the design of overcomplete dictionaries for sparse coding, Neural Gas for Dictionary Learning (NGDL), which uses a set of solutions for the sparse coefficients in each update step of the dictionary. In order to obtain such a set of solutions, we additionally propose the bag of pursuits method (BOP) for sparse approximation. Using BOP in order to determine the coefficients of the dictionary, we show in an image encoding experiment that in case of limited training data and limited computation time the NGDL update of the dictionary performs better than the standard gradient approach that is used for instance in the Sparsenet algorithm, or other state-of-the-art methods for dictionary learning such as the Method of Optimal Directions (MOD) or the widely used K-SVD algorithm. In an application to image reconstruction, dictionaries trained with this algorithm outperform not only overcomplete Haar-wavelets and overcomplete discrete cosine transformations (DCT), but also dictionaries obtained with widely used algorithms like K-SVD.

Keywords: Dictionary Learning, Image Reconstruction, Vector Quantization, Sparse Approximation

1. Introduction

Suppose, we are given an image that is incomplete. Our task is to reconstruct the original image, i.e., determine its missing pixel values from the the remaining pixels. Among other approaches, Wavelet representations have been successfully used to tackle this task [1, 2]. They have also been applied to the closely related problem of image denoising [3]. A key property for the success of certain Wavelet bases in these tasks is that natural image patches can be represented as a sparse linear combination of the Wavelets [4, 5]. Recently, for instance in [6, 7], it has been proposed to use dictionaries that have been learned from image data in order to solve these tasks. A possible advantage of the learned dictionaries is that they can be adapted to the specific properties of a subclass of images which is not always possible in case of Wavelet bases. However, in order to actually exploit this advantage, one needs a method for dictionary learning that is able to extract the subclass specific information from the training images.

Article published in Neurocomputing ***74(9)****, 2011, pp. 1418-1428*
http://dx.doi.org/10.1016/j.neucom.2011.02.002

A central problem in many dictionary learning algorithms is that for the update of the dictionary the hidden parameters of the underlying data model, i.e., the dictionary coefficients, have to be fixed even though there might be many different good configurations for the hidden parameters. In order to remedy this problem, we here propose a soft-competitive dictionary learning approach that can employ a large set of possible configurations of the hidden parameters in each update step of the dictionary. The update step for the dictionary is independent from the method that determines the configurations of the hidden parameters which makes it suitable for a broad range of different data models.

In order to formalize the problem of image reconstruction, we consider a set of vectors $\mathbf{p}_1, \ldots, \mathbf{p}_P$, $\mathbf{p}_i \in \mathbb{R}^D$. Each vector $\mathbf{p}_i$ contains the pixels of a patch of size $d \times d$, $d^2 = D$ at position i in the given image where P is the number of all the patches of size $d \times d$ of the original image. If some of the image pixels are missing, this can be written as

$$\mathbf{p}_i = S_i \mathbf{q}_i \ , \quad i = 1, \ldots, P \tag{1}$$

where S_i is a matrix that describes a projection to a lower-dimensional subspace. $\mathbf{q}_i \in \mathbb{R}^N$ contains the pixels of a patch of size $n \times n$, $n^2 = N$ at position i in the original image. The operator S_i might differ for each position of the image depending on the pixels that are missing.

In order to obtain the original image, one has to invert the mapping S_i. Since in the given image certain pixels are missing, $D < N$ holds, i.e., in this case linear algebra tells us that in gerneral S_i cannot be inverted. A common hypothesis is that the patches of the original image can be represented as a sparse linear combination of some dictionary C plus additive noise:

$$\mathbf{q}_i = C\hat{\mathbf{a}}_i + \boldsymbol{\epsilon}_i \Rightarrow \mathbf{p}_i = S_i C\hat{\mathbf{a}}_i + S_i\boldsymbol{\epsilon}_i \ . \tag{2}$$

$C = (\mathbf{c}_1, \ldots, \mathbf{c}_M)$, $\mathbf{c}_j \in \mathbb{R}^N$, where $\|\hat{\mathbf{a}}_i\|_0 \leq k \ll M$ and $\|S_i\boldsymbol{\epsilon}_i\| \leq \delta$ (M is the number of dictionary elements)[1]. Based on this hypothesis, it has been proposed (see [8] for review) to invert the mapping (2) by the solution of a sparse approximation problem

$$\mathbf{a}_i^{l_0} = \arg\min_{\mathbf{a}} \ \|\mathbf{a}\|_0 \quad \text{subject to } \|\mathbf{p}_i - S_i C\mathbf{a}\|_2 \leq \delta \tag{3}$$

where $C\mathbf{a}_i^{l_0}$ is the best approximation of $\mathbf{q}_i$ according to (3). It can be shown that $\|\mathbf{a}_i^{l_0} - \hat{\mathbf{a}}_i\|_2^2$ is upper-bounded by a constant that is proportional to the square of the noise level δ under the condition that $\mathbf{a}_i^{l_0}$ is sparse enough [9, 10, 8]. How sparse $\mathbf{a}_i^{l_0}$ has to be depends on the mutual coherence of the projected dictionary, i.e.,

$$H(S_i C) = \max_{1 \leq i,j \leq M, i \neq j} |(S_i\mathbf{c}_i)^T S_i\mathbf{c}_j| \ . \tag{4}$$

On the one hand a smaller mutual coherence $H(S_iC)$ permits a larger number of non-zero entries in $\mathbf{a}_i^{l_0}$, on the other hand a smaller number of non-zero entries

[1] $\|\mathbf{a}\|_0$ is equal to the number of non-zero entries of $\mathbf{a}$.

$\|\mathbf{a}_i^{l_0}\|_0$ permits a larger mutual coherence of the projected dictionary [9, 10, 8]. The more pixels are missing, i.e., the lower-dimensional the subspace obtained from projection S_i is, the larger the mutual coherence of the projected dictionary is expected to be.

Unfortunately, (3) is an NP-hard combinatorial optimization problem [11]. Therefore, it has been proposed, for instance in [12], to solve an easier problem instead, namely the following relaxed approximation problem

$$\mathbf{a}_i^{l_1} = \arg\min_{\mathbf{a}} \|\mathbf{a}\|_1 \quad \text{subject to } \|\mathbf{p}_i - S_i C\mathbf{a}\|_2 \leq \delta \,. \tag{5}$$

It can be shown that the solution of (5) well approximates the solution of (3) [13, 14] under realistic conditions. Among the methods that can be used to find a solution of (5) (for a comprehensive review see [8]) there are approaches based on linear or quadratic programming such as Basis Pursuit [15], iteratively reweighted least squares methods [16], stepwise methods as for instance the LARS-LASSO algorithm [17], iterative shrinkage methods [18] and related neuro-inspired dynamical systems [19].

In this work, we do not consider the relaxed optimization problem (5) but use a greedy approach that directly tackles (3). The greedy methods Orthogonal Matching Pursuit (OMP) [20] and Optimized Orthogonal Matching Pursuit (OOMP) [21] are guaranteed to provide solutions that are close to the best solution $\mathbf{a}_i^{l_0}$ if the number of non-zero entries in $\mathbf{a}_i^{l_0}$ is small enough. The smaller the number of non-zero entries is, the closer these methods approximate the best solution of (3) [22, 23, 9]. Also a neuro-inspired dynamical system has been proposed which uses a hard-sparseness constraint on the coefficients [24].

Of course, the properties of the mapping S_i strongly influence the quality of the solution, but in our setting it is not possible to choose the mapping S_i. Our task is closely related to the domain of compressive sensing [25] where a measurement setting is considered that tries to choose an optimal mapping S_i that supports easy reconstruction. Nevertheless, we can improve the performance of the approximation methods by choosing an appropriate dictionary C that is well adapted to the patches of the original image. For example in case of natural image patches obvious choices would be an (overcomplete) discrete cosine dictionary (DCT) [5] or an (overcomplete) Haar-wavelet dictionary (HAAR) [5]. Here, we do not want to use a predefined dictionary but we want to learn an appropriate dictionary.

Suppose that we have some knowledge about the content of the original image. In that case we can find a set of images that are similar to the original image and extract a large number of random patches $\mathbf{x}_1, \ldots, \mathbf{x}_L, \mathbf{x}_i \in \mathbb{R}^N$ of size $n \times n$ out of these similar images. We aim to learn a dictionary $C \in \mathbb{R}^{N \times M}$ that enables a sparse representation of these random patches

$$\min_C \frac{1}{L} \sum_{i=1}^{L} \min_{\mathbf{a}_i} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 \quad \text{subject to } \|\mathbf{a}_i\|_0 \leq k \,. \tag{6}$$

The joint solution of (6) with respect to C and $\mathbf{a}_1, \ldots, \mathbf{a}_L$ is a difficult non-convex optimization problem, whereas for given fixed coefficients one obtains

a simple convex minimization problem. In the past, many dictionary learning methods have been proposed that implement a two-step approach to tackle the joint optimization problem. First the coefficients are determined for some fixed dictionary, then the coefficients are considered fixed and an optimization step with respect to the dictionary is performed.

If one aims to use the learned dictionary on unknown data in order to solve some pratical problem such as image reconstruction, one is interested in its encoding performance on that unknown data. Given a fixed dictionary, there might exist several almost equally good configurations of the hidden coefficients. We claim that an optimization method that uses only a single configuration of the coefficients in the update of the dictionary can be suboptimal with respect to the encoding performance on unknown data whereas a soft-competitive update rule for the dictionary that employs a large number of different possible configurations of the coefficients can lead to better performance on unknown data.

In the image reconstruction experiments, we show that in case of a limited number of training samples and limited computation time, the soft-competitive approach outperforms other state-of-the-art approaches in terms of the obtained performance on unknown data. In order to obtain several configurations of the coefficients during the learning process, we propose the BOP method for sparse approximation which is derived from OOMP. It is important to note that though in this paper this BOP is used in order to determine the coefficients, the soft-competitive dictionary update step is independent from the method that provides the coefficients. Hence, this dictionary update approach can be combined with any sparse approximation method (that provides a set of solutions according to some data model) as long as a minimization of the mean squared error is obtained for some fixed coefficients. Note, that this is always the case within a probabilistic linear generative model where the additive noise is assumed to be Gaussian. Another advantage of the proposed approach is that it performs true online learning and that it converges even with a highly overcomplete dictionary. Furthermore, the update of the dictionary does not involve a matrix inversion or a singular value decomposition as it is the case for some other state-of-the-art methods.

Most of the methods for dictionary learning that have been proposed in the past can be separated into two categories: The methods from the first category use a regularization term which is scaling sensitive in order to enforce sparsity on the coefficients of the representation. Therefore, a constraint on the norm of the elements of the dictionary has to be introduced in order to obtain a non-trivial solution for the dictionary. Many methods from this category possess a probabilistic interpretation in terms of a maximization of the data likelihood or the posteriori probability of the learned dictionary according to some generative model.

The methods from the second category, the scaling invariant methods, use a measure of sparsity of the dictionary coefficients that is insensitive with respect to the scaling of the elements of the dictionary. The method which is proposed in this work belongs to the second category. In the following, we provide a brief

discussion of both categories.

2. Scaling sensitive dictionary learning

We call a dictionary learning method scaling sensitive if it uses a measure of sparseness whose influence can be minimized by an increase of the norm of the dictionary elements. This is for instance the case if one does not introduce explicit constraints on the number of non-zero entries of the coefficients but incorporates some scaling sensitive regularization term in the target function in order to prevent non-sparse solutions from being selected:

$$\min_{C} \sum_{i=1}^{L} \min_{\mathbf{a}_i} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 + \beta S(\mathbf{a}_i) \; . \tag{7}$$

The user-defined parameter β controlls the trade-off between representation error and sparseness of the coefficients. Popular scaling sensitive choices for the regularization term $S(\mathbf{a})$ are for instance:

$$S(\mathbf{a}) = \begin{cases} \|\mathbf{a}\|_1 \\ \log(1 + \mathbf{a}^T\mathbf{a}) \; . \end{cases} \tag{8}$$

The nested optimization problem (7) can be tackeled by a two-fold optimization proccess. First the dictionary C is fixed and the coefficients of the dictionary are determined by some optimization method. Then the coefficients are fixed and an optimization step with respect to the dictionary is performed.

The Sparsenet algorithm [26, 27] implements such a two-fold optimization process. First, for a fixed dictionary C, the coefficients are determined by gradient descend, which is possible if the regularization term that is used is differentiable as it is the case if $S(\mathbf{a}) = \log(1 + \mathbf{a}^2)$. After the coefficients have been determined, a gradient descent with respect to the dictionary is performed. With learning rate η, the update of the dictionary is

$$\Delta C = \eta \left(X - CA\right) A^T \; , \tag{9}$$

where $A = (\mathbf{a}_1, \ldots, \mathbf{a}_L)$ are the coefficients that have been determined before. A problematic aspect is that the gradient optimization might run into local minima if $S(\mathbf{a})$ is not convex, which is the case for $S(\mathbf{a}) = \log(1 + \mathbf{a}^2)$.

A convex optimization problem with respect to the coefficients is obtained if $S(\mathbf{a}) = \|\mathbf{a}\|_1$ is used as a regularization term. However, the optimization cannot be performed by gradient descent in this case, since the target function is not differentiable anymore. Instead, all the relaxation methods for sparse approximation that already have been mentioned can be used in order to determine the coefficients.

In the Sparsenet algorithm, the related algorithms proposed in [28, 29], or in the column normalized FOCUSS variant for dictionary learning, i.e., FOCUSS-CNDL [30], the update of the dictionary is first performed as in an unconstrained

optimization, and then the dictionary is projected to the constrained solution space by setting the norm of the dictionary elements to a fixed value. Since this approach often leads to slow convergence, improved optimization approaches have been proposed [31] that directly tackle the constrained optimization problem by an unconstrained optimization of the Langrangian dual of the target function.

Let $u > 0$ be some constant, then the constraint optimization problem for given fixed coefficients $\mathbf{a}_i$ is

$$\min_C \sum_{i=1}^{L} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 \text{ subject to } \|\mathbf{c}_j\|_2^2 \leq u \; . \tag{10}$$

The corresponding lagrangian is,

$$\begin{aligned} L_{\text{CNDL}} &= \sum_{i=1}^{L} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 + \sum_{j=1}^{M} \lambda_j (\|\mathbf{c}_j\|_2^2 - u) \\ &= trace((X - CA)^2) + trace(\Lambda(C^2 - uI)) \; , \end{aligned} \tag{11}$$

where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_M)$, $\Lambda = diag(\boldsymbol{\lambda})$ are the Lagrangian parameters. The unconstrained minimization of (11) with respect to C and $\boldsymbol{\lambda}$ is equivalent to the solution of the constrained minimization problem (10). In order to minimize (11), one considers its derivative with respect to C:

$$\frac{\partial L_{\text{CNDL}}}{\partial C} = -2(X - CA)A^T + 2C\Lambda \; . \tag{12}$$

A global minimum for any Λ is obtained at

$$\begin{aligned} \frac{\partial L_{\text{CNDL}}}{\partial C} &= 0 \\ \Leftrightarrow C &= XA^T(AA^T + \Lambda)^{-1} \; . \end{aligned} \tag{13}$$

Inserting (13) in (11), one obtains the Lagrangian dual which shall be maximized with respect to Λ. This can be done for instance by gradient descent or Newtons method. If the maximum of the Lagrangian dual with respect to Λ, i.e., Λ_*, has been determined (see [31] for details) the dictionary is obtained as

$$C = XA^T(AA^T + \Lambda_*)^{-1} \; . \tag{14}$$

The determination of the coefficients and the update of the dictionary according to (14) are subsequently repeated until some stopping criterion is met or a maximum number of learning iterations has been performed. The update (14) can be understood as the scaling sensitive variant of the Method of Optimal Directions [32], which is discussed in section 3.2.

3. Scaling invariant dictionary learning

We call a dictionary learning method scaling invariant if it uses a measure of sparsity whose influence can not be minimized by an increase of the norm of the dictionary elements. This is the case if one aims to minimize the representation error where constraints on the zero norm of the coefficients are explicitly given, i.e., one tackles the following optimization problem

$$\min_{C} \frac{1}{L} \sum_{i=1}^{L} \min_{\mathbf{a}_i} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 \quad \text{subject to } \|\mathbf{a}_i\|_0 \leq k \; . \tag{15}$$

The number M of dictionary elements and the maximum number of non-zero entries k are user-defined model parameters. Examples from this category of methods are for instance the Method of Optimal Directions (MOD) [32], the K-SVD algorithm (KSVD) [6], the Sparse Coding Neural Gas algorithm (SCNG) [33, 34], and the methods that are described in this work.

Vector quantization approaches such as the k-means algorithm [35], the LBG-algorithm [36], and the Neural Gas algorithm [37, 38, 39] are closely related to methods from this category. The vector quantization methods consider the problem (15) for $k = 1$ where an additional constraint on the coefficients has been added that enforces the coefficients to be chosen from $\{0, 1\}$.

The nested optimization problem (15) can be tackled using the same two-fold optimization approach that was already discussed for the scaling sensitive methods. In order to determine the coefficients, greedy methods such as MP [40], OMP [20], or OOMP [21] are used.

Due to the explicit constraint on the number of non-zero entries of the coefficients, the optimization with respect to the dictionary itself is unconstrained. By convention, the norm of the elements of the dictionary is set to some constant value (one) after the update of the dictionary has been performed, though this is not required in order to exclude trivial solutions. Of course, the dictionary update mechanism of the scaling invariant approaches can also be used if a scaling sensitive measure of sparseness is used, i.e., either as part of the target function, or implicitly in the method that determines the coefficients. In this case, the dictionary has to be projected to the constrained solution space by a subsequent normalization of the elements of the dictionary.

3.1. Vector quantization

The most basic method that provides sparse coding is vector quantization. Vector quantization aims to find a dictionary, i.e., codebook, C that minimizes the mean of the squared reconstruction error

$$E = \frac{1}{L} \sum_{i=1}^{L} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 \tag{16}$$

where the coefficients are subject to the constraint

$$(\mathbf{a}_i)_k = \begin{cases} 1 & : \quad k = \arg\min_j \|\mathbf{x}_i - \mathbf{c}_j\|_2^2 \\ 0 & : \quad \text{else} \; . \end{cases} \tag{17}$$

The k-means algorithm [35] is a batch method that can be used in order to determine a codebook that minimizes (16) if the coefficients are constrained by (17). In this algorithm, for each given sample $\mathbf{x}_i$, the coefficients are determined according to (17). Now let L_l be the number of given samples $\mathbf{x}_i$ with $(\mathbf{a}_i)_l = 1$. Considering the derivative of (16) with respect to the element $\mathbf{c}_l$

$$\frac{\partial E}{\partial \mathbf{c}_l} = \sum_{\mathbf{x}_i:(\mathbf{a}_i)_l=1} 2\mathbf{c}_l - \mathbf{x}_i = L_l \mathbf{c}_l - \sum_{\mathbf{x}_i:(\mathbf{a}_i)_l=1} \mathbf{x}_i \ . \tag{18}$$

A local minimum of (16) is obtained at

$$\frac{\partial E}{\partial \mathbf{c}_l} = 0 \tag{19}$$

$$\Leftrightarrow \mathbf{c}_l = \frac{1}{L_l} \sum_{\mathbf{x}_i:(\mathbf{a}_i)_l=1} \mathbf{x}_i \ . \tag{20}$$

After each element of the codebook has been subsequently updated according to (20), new coefficients are determined according to (17). The order in which the codebook vectors are updated is not relevant, since due to the constraint on the coefficients, which ensures that each sample is encoded by only one codebook vector, the update of one codebook vector does not influence the encoding performance of those samples where it has not been used. The update of the codebook and the determination of the coefficients are repeated until some stopping criterion is met. In the most common implementation the algorithm stops if the change of the codebook is lower than some threshold.

The LBG-algorithm [36] is a pattern-by-pattern method for vector quantization. At each iteration i, it considers one given sample $\mathbf{x}_i$ whose coefficients $\mathbf{a}_i$ with respect to some codebook C have been determined according to (17). At iteration i only the codebook element $\mathbf{c}_l$ with $(\mathbf{a}_i)_l = 1$ is updated. The update is computed from the gradient of $\|\mathbf{x}_i - C\mathbf{a}_i\|_2^2$ with respect to $\mathbf{c}_l$:

$$\Delta \mathbf{c}_l = \alpha \left(\mathbf{x}_i - \mathbf{c}_l \right) \ . \tag{21}$$

where α is some learning rate. If the learning rate is slowly reduced over time, the LBG algorithm performs a stochastic gradient descent on (16). Due to its stochastic nature it is more likely to reach a global minimum.

3.2. Method of optimal directions

The Method of Optimal Directions (MOD) [32] can be interpreted as a generalization of the k-means algorithm. It considers data representations that represent a given sample as an arbitrary linear combination of at most k dictionary elements, i.e, the method is applied to (15) for $k \geq 1$.

In the first step of the MOD algorithm, dictionary coefficients $A = (\mathbf{a}_1, \ldots, \mathbf{a}_L)$ of the training samples are determined by a sparse approximation method such as for instance OOMP. The dictionary can be initialized with randomly selected data samples.

In order to improve the dictionary such that the representation error is reduced, the gradient of

$$E = \|X - A\|_F^2 \tag{22}$$

with respect to the dictionary C is considered:

$$\frac{\partial E}{\partial C} = -2(X - CA)A^T \ . \tag{23}$$

A global minimum is obtained at

$$\begin{aligned} \frac{\partial E}{\partial C} &= 0 \\ \Leftrightarrow C &= XA^T(AA^T)^{-1} \ . \end{aligned} \tag{24}$$

After the dictionary has been updated according to (24) new coefficients are determined. The update of the dictionary and the determination of the coefficients are subsequently repeated until some stopping criterion is met or a maximum number of update steps have been performed.

MOD can be understood as the scaling invariant counterpart of the method proposed in [31] which can be seen from a comparison of the constrained update rule (14) and the MOD update rule (24).

3.3. K-SVD

The K-SVD algorithm is even more similar to the k-means algorithm and has been proposed in order to improve on the MOD method. According to the authors of [6] it provides better convergence speed and robustness against problematic local minima compared to the MOD method. Given fixed coefficients that have been determined by some method, in the K-SVD approach a separate update of each dictionary element is subsequently performed. Since in this case an update of a single dictionary element depends on those updates that already have been performed, the order of the updates is chosen randomly.

For the update of the l-th dictionary element only the data samples that use this element in the encoding are selected. Let $S_l = \{\mathbf{x}_i \mid (\mathbf{a}_i)_l \neq 0\}$ be the set of samples where the l-th dictionary element has been used. Now the representation error is considered that is obtained for all $\mathbf{x}_i \in S_l$ if the l-th dictionary element is removed from the encoding:

$$E_l = \sum_{\mathbf{x}_i \in S_l} \|\mathbf{x}_i - C\hat{\mathbf{a}}_i\|_F^2 = \|R_l\|_F^2 \tag{25}$$

where

$$(\hat{\mathbf{a}}_i)_m = \begin{cases} 0 & : \text{ if } \ m = l \\ (\mathbf{a}_i)_m & : \text{ else.} \end{cases} \tag{26}$$

Here R_l is a matrix that contains all the residuals that are obtained if the coefficients of the l-th dictionary element are set to zero. In order to choose a

better dictionary element, i.e., to minimize E_l, a singular value decomposition of the matrix R_l is performed:

$$R_l = U_l \Sigma_l V_l \ . \tag{27}$$

Let Σ_l be a diagonal matrix that contains the singular values of R_l. Furthermore, let λ_{l_*} be the singular value that has the largest absolute value. The updated l-th dictionary element is the column of U_l that corresponds in (27) to this largest singular value. After the l-th dictionary element has been updated also its coefficients have to be modified, since they are likely to be used in subsequent updates of those elements of the dictionary that still have to be updated. The updated coefficients of the l-th dictionary element are the entries of the l_*-th row of V_l multiplied by the largest singular value λ_{l_*}. In each iteration of the K-SVD algorithm all the dictionary elements $\mathbf{c}_l$, $l = 1, \ldots, M$ are updated this way in a random order. Then the coefficients are re-determined. This is repeated until some stopping criterion is met or a maximum number of update iterations have been performed.

4. Soft-competitive dictionary learning

All the methods that are described in section 2 and section 3 perform hard-competitive dictionary learning. Hard-competitive means that, for a given fixed dictionary, only one solution for the coefficients is used in order to compute the update of the dictionary. In the maximum likelihood interpretation of methods such as the Sparsenet algorithm [26, 27] this corresponds to an approximation of the data likelihood of the model by its maximum value in order to avoid an intractable integration over the hidden parameters of the model. In [28, 29] it has been proposed to use a better approximation of the problematic integral of the hidden parameters that is based on the gaussian integral.

Soft-competitive dictionary learning can be understood as another way of using a better approximation of the data-likelihood of the model. Instead of one "best" solution for the coefficients that is used in each update step, a certain subset of "good" solutions is used.

A pattern-by pattern soft-competitive approach for vector quantization is the Neural Gas (NG) algorithm [37, 38]. At iteration i it considers all possible encodings of a randomly selected sample $\mathbf{x}_i$. This means that the possible choices $\mathbf{a}_i^1, \ldots, \mathbf{a}_i^M$ with $(\mathbf{a}_i^j)_j = 1$ for the coefficients where $\mathbf{a}_i^j$ uses the j-th codebook vector to represent $\mathbf{x}_i$ are sorted according to the obtained reconstruction error:

$$\|\mathbf{x}_i - C\mathbf{a}_i^{j_0}\| \leq \|\mathbf{x}_i - C\mathbf{a}_i^{j_1}\| \leq \cdots \leq \|\mathbf{x}_i - C\mathbf{a}_i^{j_p}\| \leq \cdots \leq \|\mathbf{x}_i - C\mathbf{a}_i^{j_M}\| \ . \tag{28}$$

Now, in each learning iteration every codebook vector $\mathbf{c}_l$, $l = 1, \ldots, M$ is updated. Introducing the neighborhood $h_{\lambda_i}(v) = e^{-v/\lambda_i}$, the update is weighted according to the rank of the encoding that uses the codebook vector $\mathbf{c}_l$

$$\Delta\mathbf{c}_l = \alpha_t h_{\lambda_i}(\mathrm{rank}(\mathbf{x}_i, \mathbf{a}_i^l, C))\,(\mathbf{x}_i - \mathbf{c}_l) \tag{29}$$

where α_i is an exponentially decreasing learning rate

$$\alpha_i = \alpha_0 \left(\frac{\alpha_{\text{final}}}{\alpha_0} \right)^{\frac{i}{i_{\max}}} , \tag{30}$$

λ_i an exponentially decreasing neighborhood-size

$$\lambda_i = \lambda_0 \left(\frac{\lambda_{\text{final}}}{\lambda_0} \right)^{\frac{i}{i_{\max}}} , \tag{31}$$

and $i_{\max}$ the maximum number of learning iterations. It has been shown in [38] that this type of update is equivalent to a gradient descent on a well-defined cost function which for $\lambda_i \to 0$ becomes equal to (16) where the coefficients are chosen according to (17). The update rule (29) can be seen as the soft-competitive variant of the LBG update rule (21) where additionally an exponentially decreasing learning rate has been added.

In [34], we proposed the SCNG algorithm which uses a similar update rule in order to perform soft-competitive learning of the dictionary. However, the SCNG algorithm performs the updates of the dictionary in a sequence of orthogonal subspaces, since it implements an Optimized Orthogonal Matching Pursuit that modifies the dictionary during the pursuit.

Here, we want to more directly apply the NG ranking approach to sparse coding. An advantage of this more direct approach is that it can be combined with an arbitrary method for the determination of the coefficients. Similar to the NG, for each given sample $\mathbf{x}_i$ we consider all K possible coefficient vectors $\mathbf{a}_i^j$, i.e., encodings that have at most k non-zero entries. Note that K grows exponentially with M and k. The elements of each $\mathbf{a}_i^j$ are chosen such that $\|\mathbf{x}_i - C\mathbf{a}_i^j\|$ is minimal. We order the coefficients according to the representation error that is obtained by using them to approximate the sample $\mathbf{x}_i$

$$\|\mathbf{x}_i - C\mathbf{a}_i^{j_0}\| < \|\mathbf{x}_i - C\mathbf{a}_i^{j_1}\| < \cdots < \|\mathbf{x}_i - C\mathbf{a}_i^{j_p}\| < \cdots < \|\mathbf{x}_i - C\mathbf{a}_i^{j_K}\| . \tag{32}$$

If there are coefficient vectors that lead to the same reconstruction error

$$\|\mathbf{x}_i - C\mathbf{a}_i^{m_1}\| = \|\mathbf{x}_i - C\mathbf{a}_i^{m_2}\| = \cdots = \|\mathbf{x}_i - C\mathbf{a}_i^{m_V}\| , \tag{33}$$

we randomly pick one of them and do not consider the others. Note that we need this due to theoretical considerations while in practice this situation almost never occurs. Let $\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C) = p$ denote the number of coefficient vectors $\mathbf{a}_i^m$ with $\|\mathbf{x}_i - C\mathbf{a}_i^m\| < \|\mathbf{x}_i - C\mathbf{a}_i^j\|$. we consider the following modified error function

$$E_s = \sum_{i=1}^{L} \sum_{j=1}^{K} h_{\lambda_t}(\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C)) \|\mathbf{x}_i - C\mathbf{a}_i^j\|_2^2 , \tag{34}$$

which becomes equal to (16) for $\lambda_t \to 0$. In order to minimize (34), we consider the gradient of E_s with respect to C, which is

$$\frac{\partial E_s}{\partial C} = -2 \sum_{i=1}^{L} \sum_{j=1}^{K} h_{\lambda_t}(\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C))(\mathbf{x}_i - C\mathbf{a}_i^j){\mathbf{a}_i^j}^T + R \tag{35}$$

with

$$R = \sum_{i=1}^{L} \sum_{j=1}^{K} h'_{\lambda_t}(\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C)) \frac{\partial \text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C)}{\partial C} \|\mathbf{x}_i - C\mathbf{a}_i^j\|_2^2. \tag{36}$$

In order to show that $R = 0$, we adopt the proof given in [38] to our setting. With $\mathbf{e}_i^j = \mathbf{x}_i - C\mathbf{a}_i^j$, we write $\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C)$ as

$$\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C) = \sum_{m=1}^{K} \theta((\mathbf{e}_i^j)^2 - (\mathbf{e}_i^m)^2) \tag{37}$$

where $\theta(x)$ is the heaviside step function. The derivative of the heaviside step function is the delta distribution $\delta(x)$ with $\delta(x) = 0$ for $x \neq 0$ and $\int \delta(x)dx = 1$. Therefore, we can write

$$\begin{aligned} R &= \sum_{i=1}^{L} \sum_{j=1}^{K} h'_{\lambda_t}(\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C))(\mathbf{e}_i^j)^2 \\ &\quad \cdot \sum_{m=1}^{T} ((\mathbf{a}_i^j)^T - (\mathbf{a}_i^m)^T)\delta((\mathbf{e}_i^j)^2 - (\mathbf{e}_i^m)^2) \end{aligned} \tag{38}$$

Each term of (38) is non-vanishing only for those $\mathbf{a}_i^j$ for which $(\mathbf{e}_i^j)^2 = (\mathbf{e}_i^m)^2$ is valid. Since we explicitly excluded this case, we obtain $R = 0$. Hence, we can perform a stochastic gradient descent on (34) with respect to C by applying $i = 0, \ldots, i_{\max}$ updates of C using the gradient based learning rule

$$\Delta C = \alpha_i \sum_{j=1}^{K} h_{\lambda_i}(\text{rank}(\mathbf{x}, \mathbf{a}_i^j, C))(\mathbf{x}_i - C\mathbf{a}_i^j){\mathbf{a}_i^j}^T \tag{39}$$

for a randomly chosen $\mathbf{x}_i$ where the neighborhood-size λ_i and learning rate α_i are chosen according to (31) and (30). After each update has been applied the norm of the column vectors of C is set to one. Then a new training sample $\mathbf{x}_i$ is selected, the corresponding $\mathbf{a}_i^j$ are determined, and the next update of C can be performed. We term this way of updating the dictionary Neural Gas for Dictionary Learning (NGDL).

The update (39) is closely related to the standard gradient update of the Sparsent algorithm (9). For $\lambda_0 = \lambda_{\text{final}} \to 0$, $\alpha_0 = \alpha_{\text{final}} = \eta$, and a batch size of one both update rules are the same. In constrast to K-SVD, MOD (section 3.3 and 3.2) and the scaling sensitive variant of MOD (section 2) the update (39) works in a pattern-by-pattern mode and does not invole a matrix inversion or a singular value decomposition.

5. A bag of orthogonal matching pursuits (BOP)

So far, for each training sample $\mathbf{x}$, all possible coefficient vectors $\mathbf{a}^j$, $j = 1, \ldots, K$ with $\|\mathbf{a}^j\|_0 \leq k$ have been considered. Since K grows exponentially with M and

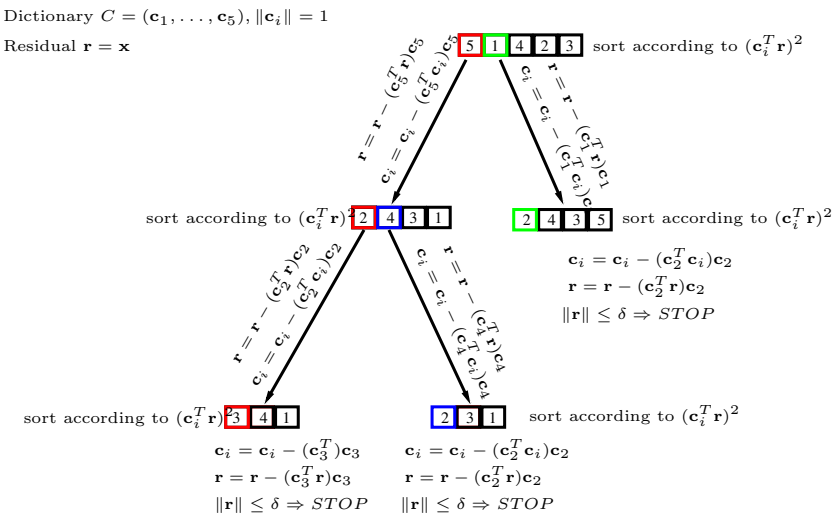

Figure 1: The figure depicts the tree-like search procedure of the BOP method. In this example $K_{\text{user}} = 3$ holds, i.e., the method determines three different solutions. The method starts by sorting the dictionary elements according to their overlap with respect to the residual (root of the tree). The dictionary element that has the largest overlap, i.e., element 5 is selected. All other dictionary elements as well as the residual are orthogonalized with respected to dictionary element 5. This procedure is repeated (elements 2 and 3 are selected) until the norm of the residual drops below the threshold δ. Now, the second solution is determined. Among all overlaps that have been computed so far the largest one is selected (element 1 at root level). Again a sequence of orthogonalizations is performed until the norm of the residual becomes small enough (after element 2 has been selected). The third solution is obtained by repeating the entire procedure again. Solution 1: 5,2,3 Solution 2: 1,2 Solution 3: 5,4,2.

k, this approach is not applicable in practice. However, since in (34) all those contributions in the sum for which the rank is larger than the neighborhood-size λ_t can be neglected, we only need the first best ones with respect to the reconstruction error.

In the following, we extend OOMP so that not only the best but the first K_{user} best coefficients $\mathbf{a}^j$ are determined, at least approximately. This is done in a tree-like search procedure which is illustrated in Figure 1. OOMP iteratively constructs a given sample $\mathbf{x}$ out of the columns of the dictionary C. The algorithm starts with $U_n^j = \emptyset$, $R_0^j = (\mathbf{r}_1^{0,j}, \ldots, \mathbf{r}_M^{0,j}) = C$ and $\boldsymbol{\epsilon}_0^j = \mathbf{x}$. The set U_n^j contains the indices of those columns of C that have been used during the j-th pursuit with respect to $\mathbf{x}$ up to the n-th iteration. R_n^j is a temporary matrix that has been orthogonalized with respect to the columns of C that are indexed by U_n^j. $\mathbf{r}_l^{n,j}$ is the l-th column of R_n^j. $\boldsymbol{\epsilon}_n^j$ is the residual in the n-th iteration of the j-th pursuit with respect to $\mathbf{x}$.

In iteration n, the algorithm looks for that column of R_n^j whose inclusion in the linear combination leads to the smallest residual $\boldsymbol{\epsilon}_{n+1}^j$ in the next iteration of the algorithm, i.e., that has the maximum overlap with respect to the current residual. Hence, with

$$\mathbf{y}_n^j = \left({\mathbf{r}_1^{n,j}}^T \boldsymbol{\epsilon}_n^j)/\|\mathbf{r}_1^{n,j}\|, \ldots, ({\mathbf{r}_l^{n,j}}^T \boldsymbol{\epsilon}_n^j)/\|\mathbf{r}_l^{n,j}\|, \ldots, ({\mathbf{r}_M^{n,j}}^T \boldsymbol{\epsilon}_n^j)/\|\mathbf{r}_M^{n,j}\| \right) \tag{40}$$

it looks for

$$l_{\text{win}}(n,j) = \arg \max_{l, l \notin U_n^j} ((\mathbf{y}_n^j)_l)^2 \, . \tag{41}$$

Then, the orthogonal projection of R_n^j to $\mathbf{r}_{l_{\text{win}}(n,j)}^{n,j}$ is removed from R_n^j

$$R_{n+1}^j = R_n^j - \left(\mathbf{r}_{l_{\text{win}}(n,j)}^{n,j} ({R_n^j}^T \mathbf{r}_{l_{\text{win}}(n,j)}^{n,j})^T \right) / ({\mathbf{r}_{l_{\text{win}}(n,j)}^{n,j}}^T \mathbf{r}_{l_{\text{win}}(n,j)}^{n,j}) \, . \tag{42}$$

Furthermore, the orthogonal projection of $\boldsymbol{\epsilon}_n^j$ to $\mathbf{r}_{l_{\text{win}}(n,j)}^{n,j}$ is removed from $\boldsymbol{\epsilon}_n^j$

$$\boldsymbol{\epsilon}_{n+1}^j = \boldsymbol{\epsilon}_n^j - \left(({\boldsymbol{\epsilon}_n^j}^T \mathbf{r}_{l_{\text{win}}(n,j)}^{n,j}) / ({\mathbf{r}_{l_{\text{win}}(n,j)}^{n,j}}^T \mathbf{r}_{l_{\text{win}}(n,j)}^{n,j}) \right) \mathbf{r}_{l_{\text{win}}(n,j)}^{n,j} \, . \tag{43}$$

The algorithm stops if $\|\boldsymbol{\epsilon}_n^j\| \leq \delta$ or $n = k$. The j-th approximation of $\mathbf{a}$, i.e., $\mathbf{a}^j$, can be obtained by recursively tracking the contribution of each column of C that has been used during pursuit j. So far, we described how a pursuit is performed. In order to obtain a set of approximations $\mathbf{a}^1, ..., \mathbf{a}^{K_{\text{user}}}$, where K_{user} is chosen by the user, we want to conduct K_{user} different pursuits. To obtain K_{user} different pursuits, we implement the following function

$$Q(l,n,j) = \begin{cases} \begin{array}{l} \text{If there is no pursuit among all pursuits that have} \\ \text{been performed with respect to } \mathbf{x} \text{ that is equal to} \\ \text{the } j\text{-th pursuit up to the } n\text{-th iteration where in} \\ \text{that iteration column } l \text{ has been selected} \end{array} \\ 0: \\ 1: \text{else} \, . \end{cases} \tag{44}$$

Additionally, we track all overlaps $\mathbf{y}_n^j$ that have been computed during a pursuit j. Let s_j be the number of iterations of the j-th pursuit. If m pursuits have

Figure 2: The set of images where the training and test patches were extracted from.

been performed, among all previous pursuits, we look for the largest overlap that has not been used so far:

$$j_{\text{target}} = \arg \max_{j=1,\ldots,m} \max_{n=0,\ldots,s_j-1} \max_{l,Q(l,n,j)=0} ((\mathbf{y}_n^j)_l)^2 \tag{45}$$

$$n_{\text{target}} = \arg \max_{n=0,\ldots,s_{j_{\text{target}}}-1} \max_{l,Q(l,n,j_{\text{target}})=0} ((\mathbf{y}_n^{j_{\text{target}}})_l)^2 \tag{46}$$

$$l_{\text{target}} = \arg \max_{l,Q(l,n_{\text{target}},j_{\text{target}})=0} ((\mathbf{y}_{n_{\text{target}}}^{j_{\text{target}}})_l)^2 \,. \tag{47}$$

We replay pursuit j_{target} up to iteration n_{target}. In that iteration, we select column l_{target} instead of the previous winner and continue with the pursuit until the stopping criterion has been reached. We repeat this procedure until K_{user} pursuits have been performed. See also Figure 1 for an illustration of the BOP method.

6. Experiments

We studied the influence of the soft-competive update of the dictionary in combination with the BOP method on the performance of the learned dictionary under the condition of a limited number of training samples and limited time being available for learning. The measure of performance was the representation error obtained on unknown data.

We randomly extracted 6000 image patches of size 8×8 from a set of 11 images of size 400×600. Most of the training images are photographs of the city of Brugge. They are depicted in Figure 2. The training patches were selected

randomly but with a variance within each patch of at least 0.1. We divided the set of random patches in a training and test part, each of size 3000. In a first experiment, we evaluated the test error,

$$E_{\mathrm{TEST}} = \frac{1}{3000} \sum_{i=1}^{3000} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 \tag{48}$$

where $\mathbf{x}_i$ are the elements of the test set, the coefficients $\mathbf{a}_i$ were determined by the BOP method with $K_{\mathrm{user}} = 1, \ldots, 17$, and C was either an overcomplete DCT dictionary of size 441 or an overcomplete HAAR dictionary of size 441. The number k of permitted non-zero coefficients was varied between $k = 3$ and $K = 13$. The results are shown in Figure 3. It can be seen that the DCT dictionary performs better than the HAAR dictionary and that the reconstruction error remains constant though the number of BOP-trials increases. Hence, in this case, there is only a slight performance gain obtained by using the BOP method instead of OOMP for the determination of the coefficients. Note, that for $K_{\mathrm{user}} = 1$ BOP is equal to OOMP.

Next, we evaluated the performance that is obtained on the test set if the dictionary has been learned on the training set. In this evalution, we included the soft-competitive approach proposed here (NGDL plus BOP soft), the scaling invariant methods for dictionary learning MOD and K-SVD that have been discussed in Section 3, and the simple gradient descent update rule of the Sparsenet algorithm (9) (GRAD), which is discussed in Section 2. In order to evaluate if soft-competitive learning actually makes a difference, we also learned a dictionary where the neighborhood-size used in the NGDL+BOP method was practically zero (denoted by NGDL plus BOP hard, $\lambda_0 = \lambda_{\mathrm{final}} = 10^{-10}$), which corresponds to hard-competitive learning. Again the parameter K_{user} was varied from $K_{\mathrm{user}} = 1$ to $K_{\mathrm{user}} = 17$ and the number of non-zero coefficients was varied between $k = 3$ and $K = 13$.

As already mentioned, we wanted to evaluate the best performance that can be obtained when both the number of training samples and the available computation time are limited. We used the training set consisting of 3000 samples of size 8×8 that has been introduced above. The computational demand of all the methods for dictionary learning that have been compared is dominated by the computational effort for the estimation of coefficients of the dictionary. Therefore, for all methods, we fixed computation time by permitting 30000 estimations of the coefficients. In case of the pattern-by-pattern method NGDL+BOP this corresponds to 10 runs over the entire training set, i.e., 10 training epochs. All the other approaches are batch methods. For a fixed number of estimations of the coefficients the number of updates of the dictionary is equal to $30000/B$ where B is the batch size. Of course, the batch size is upper-bounded by the number of training samples that are given. Using all the training data in each update step would, due to the limited number of estimations of the coefficients, correspond to only 10 updates of the dictionary. Since such a small number of dictionary updates might be suboptimal the batch size B was varied and $30000/B$ updates of the dictionary were performed. We report the results for

the best trade-off between number of updates of the dictionary and batch size. Apart from the batch size, further parameters had to be chosen for some of the methods. This was done in the following way:

- **Simple gradient (GRAD)**: For a fixed learning rate $\eta = 0.1$ the batch size was varied in 10 steps between 1 and 3000 and the obtained error on the test set was evaluated. For each tested batch size $30000/B$ updates of the dictionary were performed. The batch size was set to the value that yielded the smallest error on the test set. Using the optimal batch size, the learning rate was varied in 10 steps between 0.01 and 1.0. We report the error on the test set for the optimal batch size and learning rate.

- **NGDL+BOP soft**: For $\lambda_0 = K_{\text{user}}, \alpha_0 = 1.0, \alpha_{\text{final}} = 0.01$ the final neighborhood-size was varied in 10 steps between $\lambda_{\text{final}} = 0.001$ and $\lambda_{\text{final}} = K_{\text{user}}$. Using the optimal final neighborhood-size with respect to the error on the test set the initial neighborhood-size was varied in 10 steps between $\lambda_0 = \lambda_{\text{final}}$ and $\lambda_0 = K_{\text{user}}$. Using the optimal initial and final neighborhood-sizes the final learning rate was varied in 10 steps between $\alpha_{\text{final}} = 0.01$ and $\alpha_{\text{final}} = 1.0$. Using the optimal initial and final neighborhood-sizes and the optimal final learning rate, the initial learning rate was varied in 10 steps between $\alpha_0 = \alpha_{\text{final}}$ and $\alpha_0 = 1.0$. We report the error on the test set that is obtained for optimal initial and final learning rate and neighborhood-size. Note that for $k > 3$ the optimal final neighborhood-size was always the largest tested value, i.e., $\lambda_{\text{final}} = K_{\text{user}}$. This was also the case for $k = 3$ for $K_{\text{user}} < 17$ while in case of $k = 3$ and $K_{\text{user}} = 17$ as optimal final neighborhood-size $\lambda_{\text{final}} = 11.9$ was selected.

- **NGDL+BOP hard**: The same initial and final learning rates were used that were optimal in the soft-competitive case. The initial- and final neighborhood-sizes were set to $\lambda_0 = \lambda_{\text{final}} = 10^{-10}$.

- **MOD**: The batch size B was varied in 10 steps between 442 and 3000. For each tested batch size $30000/B$ updates of the dictionary were performed. We report the smallest error on the test set over all tested batch sizes.

- **K-SVD**: The batch size was varied in 10 steps between 442 and 3000. For each tested batch size $30000/B$ updates of the dictionary were performed. We report the smallest error on the test set over all tested batch sizes.

For all methods except NGDL+BOP only the coefficients of the best solution provided by BOP were used for learning. In order to remove the DC-component from the image patches, for all methods the first dictionary element was set to $\sqrt{1/64}$ and kept constant during learning. It was also forced to participate in each linear combination that was determined by the BOP method.

The results for the learned dictionaries are shown in Figure 4. It can be seen that in contrast to the DCT and HAAR dictionaries the performance can be improved by increasing the number of BOP trials (K_{user}). NGDL+BOP in soft-competitive mode performs best while K-SVD performs second best.

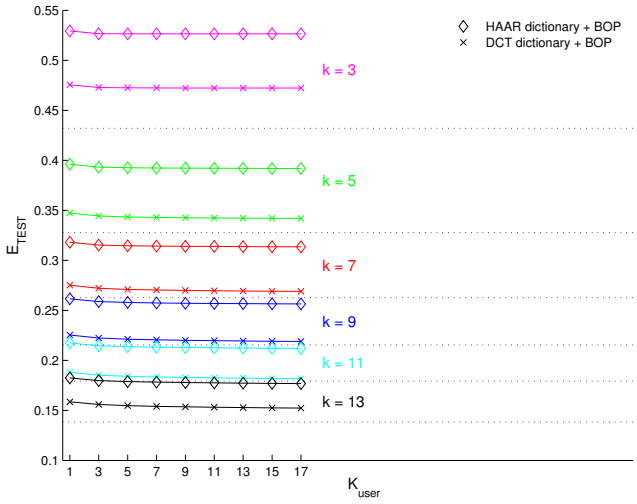

Figure 3: Mean squared representation error of the 3000 patches of size 8×8 that were randomly extracted from the images that are depicted in Figure 2. Either an overcomplete DCT dictionary or an overcomplete HAAR dictionary were used for the encoding. In this case, the BOP method provides only a slight improvement. Note that for $K_{\text{user}} = 1$ BOP is equal to OOMP.

NGDL+BOP in hard-competitive mode performs as good as the standard gradient descent approach when using a constant learning rate (GRAD+BOP). It has been shown in [41] that in certain circumstances K-SVD can be outperformed by the gradient descend approach, however, in our experiment the performance of K-SVD is slightly better. We tested whether the performance gap between K-SVD+BOP and NGDL+BOP in soft-competitive mode can be reduced by performing $10\times$ more learning iterations. Therefore, also results for 100 learning epochs with $K_{\text{user}} = 17$ for K-SVD+BOP and NGDL+BOP in soft-competitive mode are reported. It can be seen from (4) that the gap is reduced but still noticeable.

6.1. Image reconstruction

In the last experiment, we evaluated the influence of the soft-competitive learning on the performance in an image reconstruction task. In this experiment we used those methods that performed best in the previous experiment, i.e., K-SVD+BOP and NGDL+BOP.

We created a larger training set, which better resembles the statistics of the images to be reconstructed. This larger set consists of 150000 image patches that were extracted randomly from the images that are depicted in Figure 2. For all methods, we used the optimal parameters that were determined in the previous experiment except for K-SVD where we increased the batch size by a factor of 3. We performed only 1 learning epoch which means that each training sample was presented exactly once to the learning algorithm. However, due to the much

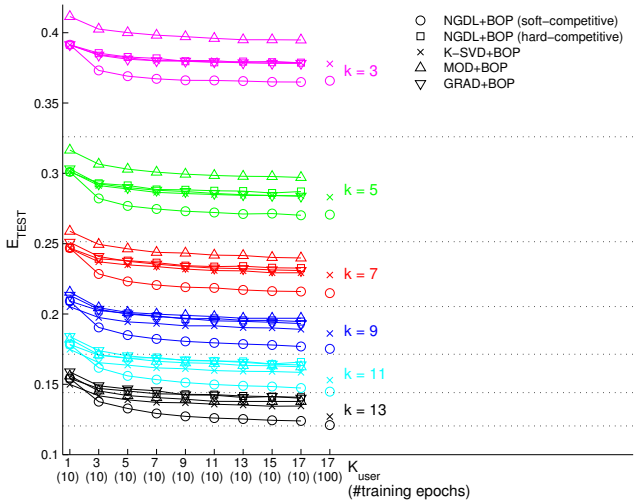

Figure 4: Mean squared representation error of the 3000 patches of size 8×8 of the test set. Results for varying number of permitted non-zero coefficients $k = 3, \ldots, 13$ are shown. MOD+BOP: Dictionary learned with the Method of Optimal directions. K-SVD+BOP: Dictionary learned with the K-SVD algorithm. NGDL+BOP(soft-competitive): Method proposed in this paper in soft-competitive mode. NGDL+BOP(hard-competitive): Method proposed in this paper in hard-competitive mode. GRAD+BOP: Gradient descent with constant learning rate combined with BOP. Note that for $K_{\text{user}} = 1$ BOP is equal to OOMP. It can be seen that for all methods the performance improves if K_{user} is increased. NGDL+BOP in soft-competitive mode performs best, K-SVD second best. For all methods the user-defined parameters were optimized (see text for details). All methods used a separate training set.

larger training set this corresponds to an increase of the number of dictionary updates by a factor of ≈ 2 in case of the K-SVD algorithm. Again, the number of permitted non-zero coefficients was varied between $k = 3$ and $k = 13$. We set K_{user} to the value that performed best in the previous experiment which is 17 as can be seen from Figure 4. The computation time for each k and method are shown in Table 5.

We used 78 test images of size 400×600 which are similar to the images depicted in Figure 2. From each test image, we removed certain percentages of pixels (0%,30%, 50%, 70%, 90%). Then, for each 8×8 patch of the incomplete images we computed the coefficients with respect to a given dictionary using the BOP method ($K_{\text{user}} = 17$). Only the remaining pixels were used, i.e., the dimensions of the dictionary elements corresponding to the missing pixels were not considered. This corresponds to solving (3) where S_i is a projection into a low-dimensional subspace.

The minimum norm of the residual for the BOP stopping criterion, δ, was varied between $0.00032 = \|0.00004 \cdot \mathbf{1}\|$, $0.0032 = \|0.0004 \cdot \mathbf{1}\|$,$0.032 = \|0.004 \cdot \mathbf{1}\|$,$\mathbf{1} \in \mathbb{R}^{64}$ which corresponds to an average error of approximately ± 0.01, ± 0.1, and ± 1 in 8-bit grayscale images. Note that δ was dynamically adjusted according to the number of pixels that are actually present in an image patch, i.e., a patch containing n pixels was approximated using a minimum norm $\delta_n = \delta\sqrt{64/n}$. In order to reconstruct the image patch, we took the linear combination of the complete dictionary elements using the coefficients that minimize the norm of the residual. We performed the estimation of the coefficients only for non-overlapping patches since this is computationally more efficient. Of course, the reconstruction can be improved by estimating the coefficients for each possible patch of the image and using the mean value of all estimated patches at a certain position as final estimator for the pixel value at that position. However, this is not required for the comparison of the performance of different methods for dictionary learning.

For each setting of the number of permitted non-zero coefficients k and each choice of the noise level parameter δ and for each method we computed the mean PSNR value over all 78 test images. In Table 1 the best mean PSNR value for different percentages of missing pixels and different methods are shown.

It can be seen that for all percentages of missing pixels the NGDL+BOP soft-competitive approach provides the best mean PSNR value for a certain choice of k and δ. Which k and δ the best choice are, can be seen from Table 3 and Table 4. From Table 3, which lists the best choice for the permitted number of non-zero entries it can be seen that the more pixels are missing the less coefficients should be estimated. From Table 4 it follows that the noise level δ should be chosen smaller as more pixels are available for the estimation of the coefficients.

The test images are quite different and, therefore, the standard deviation of the PSNR values over the 78 test images is around 4. Also the differences of the mean PSNR values are not that large. Therefore, we took a closer look at the results and counted for each percentage of missing pixels how often each method provided the best PSNR value over all 78 test images. The result is shown in

percent missing	DCT	HAAR	NGDL+BOP (soft)	NGDL+BOP (hard)	K-SVD
0%	37	36.3	39.6	38.9	39.1
30%	31	28.4	32.9	32.5	32.4
50%	27.8	25.6	29.7	29.6	29.6
70%	24.6	23.4	26.8	26.7	26.6
90%	17.5	17.9	18.1	18.1	18.1

Table 1: Best mean PSNR values over 78 test images for a certain choice of k, the number of permitted non-zero entries, and δ, the noise level parameter used as stopping criterion in the reconstruction of the images. Which choices of k and δ correspond to these mean PSNR values can be seen from Table 3 and Table 4. The dictionaries obtained with NGDL+BOP(hard/soft) and K-SVD+BOP that performed best in case of 0% missing pixels are depicted in Figure 5.

percent missing	DCT	HAAR	NGDL+BOP (soft)	NGDL+BOP (hard)	K-SVD+BOP
0%	0	0	77	0	1
30%	0	0	69	9	0
50%	0	0	52	21	5
70%	0	0	57	15	6
90%	0	13	11	36	18

Table 2: For each percentage of missing pixels, the table shows how often a certain method provided the best PSNR value over the 78 test images. Note, that the best k and δ values were not chosen separately for each image, but are exactly the same over all images according to Table 3 and Table 4.

Table 2. The best k and δ values were not selected for each image separately but are exactly the same over all images according to Table 3 and Table 4. It can be seen from Table 2 that the soft-competitive approach not only provides the best mean PSNR value but that its performance is best for the majority of the test images. The more pixels are missing, the less clear the result is. For 90% missing pixels all methods perform equally bad. Those dictionaries learned with NGDL+BOP and K-SVD+BOP that perform best in the image encoding task, i.e., where no pixels are missing, are depicted in Figure 5.

7. Conclusion

We proposed the novel bag of pursuits (BOP) method for sparse approximation which determines a set of configurations of the dictionary coefficients where each configuration leads to a small reconstruction error with respect to the approximated sample. This is in constrast to many other sparse approximation methods which provide just a single solution to the approximation problem. This novel approach for sparse approximation is combined with a novel approach for soft-competitive pattern-by-pattern learning of dictionaries (NGDL) that is derived from the Neural Gas (NG) vector quantization method.

BOP and NGDL address a central problem in many dictionary learning algorithms, namely, that for the update of the dictionary the hidden parameters

percent missing	DCT	HAAR	NGDL+BOP (soft)	NGDL+BOP (hard)	K-SVD+BOP
0%	13	13	13	13	13
30%	11	13	11	11	11
50%	5	3	5	5	5
70%	3	3	3	3	3
90%	5	7	3	3	3

Table 3: Best choice for k, the number of permitted non-zero entries in the estimation of the dictionary coefficients. For instance, in case of 30% missing pixels NGDL+BOP(soft) achieves the best mean PSNR value over all 78 test images with $k = 11$ non-zero coefficients.

percent missing	DCT	HAAR	NGDL+BOP (soft)	NGDL+BOP (hard)	K-SVD+BOP
0%	0.00004	0.0040	0.00004	0.00004	0.00004
30%	0.0004	0.0040	0.00004	0.00004	0.0004
50%	0.0004	0.0040	0.0004	0.0004	0.0004
70%	0.0040	0.0040	0.0040	0.0040	0.0040
90%	0.0040	0.0040	0.0040	0.0040	0.0040

Table 4: Best choice for δ, the noise level parameter that is used as stopping criterion in the estimation of the coefficients for reconstruction of the images. For instance, in case of 70% missing pixels NGDL+BOP(soft) achieves the best mean PSNR value over all 78 test images with $\delta = 0.0040$ and $k = 3$ (see Table 3).

k	NGDL+BOP (soft)	NGDL+BOP (hard)	K-SVD+BOP
3	20	19.5	21.8
5	37.7	36.7	38.3
7	54.6	53.4	56.4
9	75	72.4	74.7
11	92	97.4	99.8
13	120.7	115.1	120.5

Table 5: Computation time in minutes on a recent desktop computer for each number of permitted non-zero coefficients and method. For all method 150000 coefficient estimations were performed. For all methods the coefficients were determined with BOP ($K_{\text{user}} = 17$)

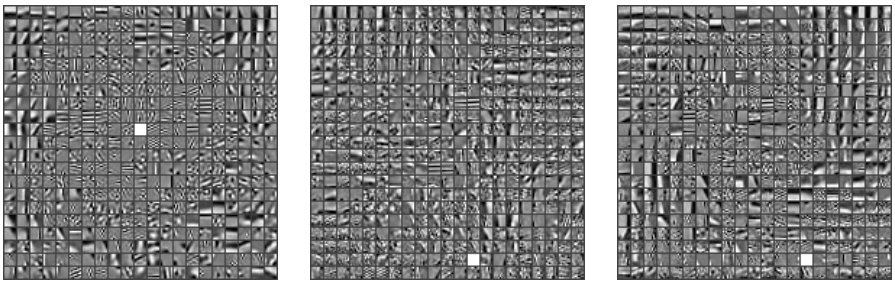

Figure 5: The dictionaries for the image reconstruction obtained with $K_{\text{user}} = 17$ and $k = 13$ permitted non-zero coefficients. **Left:** NGDL+BOP soft-competitive ($\lambda_0 = \lambda_{\text{final}} = 17$), **Center:** NGDL+BOP hard-competitive($\lambda_0 = \lambda_{\text{final}} = 10^{-10}$), **Right:** K-SVD+BOP. The dictionary size is 441. The training set consisted of 150000 randomly extracted image patches of size 8×8. The patches were extracted from 11 images of size 400×600, which are shown in Figure 2.

of the underlying data model, i.e., the dictionary coefficients, have to be fixed even though there can be many different configurations of the hidden parameters that have a high probability according to the model. In the maximum likelihood interpretation of methods such as the Sparsenet algorithm [26, 27] this corresponds to an approximation of the data likelihood of the model by the maximum value of the likelihood function. This approximation is used in order to avoid an intractable integration over the hidden parameters of the model. Improved methods that try to use a better approximation of the problematic integral have been proposed in [28, 29]. NGDL can be seen as a novel way of dealing with this problem. In contrast to many state-of-the-art methods for dictionary learning, it can use a large set of possible configurations of the hidden parameters in each update step of the dictionary.

NGDL is more versatile than the previously introduced sparse-coding-neural-gas (SCNG) method for soft-competitive dictionary learning, since it can be combined with different sparse approximation methods. For instance, it can be applied in order to learn the dictionary whenever a probabilistic linear generative model is used where the additive noise is assumed to be Gaussian. However, in future work, new methods have to developed that can be used to determine a set of solutions for the coefficients in order to apply NGDL to such problems which often do not use the L_0 norm as measure of sparseness.

NGDL is powerful because it avoids matrix inversion or singular value decomposition and, due to its stochastic nature, should not so easily get trapped in local minima. The potential of our novel approach is illustrated by image-reconstruction results, which demonstrate that the soft-competitive learning version can outperform a number of state-of-the-art methods.

References

[1] M. Antonini, M. Barlaud, Pierre Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Trans. on Image Proccess.*, 1(2):205–220,

1992.

[2] J. L. Starck, A. Bijaoui, B. Lopez, and C. Perrier. Image reconstruction by the wavelet transform applied to aperture synthesis. *Astron and Astrophysics*, 283(1):349–360, 1994.

[3] S. G. Chang, Y. Bin, and M. Vetterli. Adaptive wavelet thresholding for image denoising and compression. *IEEE Trans. on Image Process.*, 9(9):1532–1546, 2000.

[4] Alfred M. Bruckstein, David L. Donoho, and Michael Elad. From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images , 2007.

[5] Stephane Mallat. *A Wavelet Tour of Signal Processing, 3rd ed., Third Edition: The Sparse Way*. Academic Press, 3 edition, December 2008.

[6] 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.

[7] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *IEEE Trans on Image Proccess.*, 17(1), 2008.

[8] A. M. Bruckstein, D. L. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81, 2009.

[9] 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.

[10] R. Gribonval, R. Figueras, and P. Vandergheynst. A simple test to check the optimality of a sparse signal approximation. *Signal Process.*, 86:496–510, 2006.

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

[12] E. J. Candés, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Comm. Pure Appl. Math.*, 59:1207–1233, 2006.

[13] D. L. Donoho. For most large underdetermined systems of linear equations, the minimal l_1-norm near-solution approximates the sparsest near solution. *Commun. Pure Appl. Math.*, 59:907–934, 2006.

[14] D. L. Donoho. For most large underdetermined systems of linear equations, the minimal l_1-norm solution is also the sparsest solution. *Commun. Pure Appl. Math.*, 59:797–829, 2006.

[15] Scott S. Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Rev.*, 43(1):129–159, 2001.

[16] B. D. Rao, K. Engan, S. F. Cotter, J. Palmer, and K. Kreutz-Delgado. Subset selection in noise based on diversity measure minimization. *IEEE Trans. Signal Process.*, 51:760–770, 2003.

[17] M. R. Osborne, B. Presnell, and B. A. Turlach. A new approach to vairable selection in least squares problems. *J. Numer. Anal.*, 20:389–403, 2000.

[18] I. Daubechies, M. Defrise, and C.De-Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Commun. Pure Appl. Math.*, 57:1413–1457, 2004.

[19] C. J. Rozell and D. H. Johnson R. G. Baraniuk. Sparse coding via thresholding and local competition in neural circuits. *Neural Computation*, 20:2526–2563, 2008.

[20] 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.

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

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

[23] 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.

[24] Martin Rehn and Friedrich T. Sommer. A network that uses few active neurones to code visual input predicts the diverse shapes of cortical receptive fields. *Journal of Computational Neuroscience*, 22(2):135–146, 2007.

[25] David L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[26] 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.

[27] 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.

[28] Michael S. Lewicki and Terrence J. Sejnowski. Learning nonlinear overcomplete representations for efficient coding. In *NIPS '97: Proceedings of the 1997 conference on Advances in neural information processing systems 10*, pages 556–562, Cambridge, MA, USA, 1998. MIT Press.

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

[30] 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.

[31] 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.

[32] K. Engan, S. O. Aase, and J. Hakon Husoy. Method of optimal directions for frame design. In *ICASSP '99: Proceedings of the Acoustics, Speech, and Signal Processing, 1999. on 1999 IEEE International Conference*, pages 2443–2446, Washington, DC, USA, 1999. IEEE Computer Society.

[33] 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.

[34] Kai Labusch, Erhardt Barth, and Thomas Martinetz. Sparse Coding Neural Gas: Learning of Overcomplete Data Representations. *Neurocomputing*, 72(7-9):1547–1555, 2009.

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

[36] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *Communications, IEEE Transactions on*, 28(1):84–95, 1980.

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

[38] 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.

[39] Marie Cottrell, Barbara Hammer, Alexander Hasenfuß, and Thomas Villmann. Batch and median neural gas. *Neural Netw.*, 19(6):762–771, 2006.

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

[41] G. Monaci, P. Vandergheynst, and F. T. Sommer. Learning Bimodal Structure inData. *IEEE Transactions on Neural Networks*, 20:1898–1910, 2009.