ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning. Bruges (Belgium), 28-30 April 2010, d-side publi., ISBN 2-930307-10-2.

Learning Sparse Codes for Image Reconstruction

Kai Labusch 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 that generalizes the Sparse Coding Neural Gas (SCNG) algorithm such that it is not bound to a particular approximation method for the coefficients of the dictionary elements. In an application to image reconstruction, a dictionary that has been learned using this algorithm outperforms a dictionary that has been obtained from the widely-used K-SVD algorithm, an overcomplete Haar-wavelet dictionary and an overcomplete discrete cosine transformation (DCT).

1 Introduction

In order to learn a new representation of given data $\mathbf{x}_1, \ldots, \mathbf{x}_L, \mathbf{x}_i \in \mathbb{R}^N$, we are looking for a dictionary $C \in \mathbb{R}^{N \times M}$ that minimizes

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

where $\mathbf{x}_i^{\text{opt}} = C\mathbf{a}_i$ with $\mathbf{a}_i = \arg\min_{\mathbf{a}} \|\mathbf{x}_i - C\mathbf{a}\|$, $\|\mathbf{a}\|_0 \leq k$ denotes the best k-term representation[1] of $\mathbf{x}_i$ in terms of C. The number of dictionary elements M and the maximum number of non-zero entries k are user-defined model parameters. Methods such as Optimized Orthogonal Matching Pursuit (OOMP) [1] can be used in order to find an approximation of the coefficients $\mathbf{a}_i$ of the best k-term representation. The K-SVD algorithm [2] can employ an arbitrary approximation method for the coefficients in order to learn a dictionary from the data. Using data that actually was generated as a sparse linear combination of some given dictionary, it has been shown that K-SVD or Sparse Coding Neural Gas (SCNG) [3] can be used in order to reconstruct the dictionary only from the data even in highly overcomplete settings under the presence of strong noise. Unlike K-SVD, the SCNG algorithm is bound to a specific approximation method for the coefficients, i.e., OOMP.

Here, we propose a generalization of the SCNG approach that, like K-SVD, can employ an arbitrary approximation method for the coefficients. In order to demonstrate the performance of the method, we learn a sparse dictionary for image reconstruction. We compare the obtained performance of the learned dictionary to the performance of a dictionary learned with K-SVD, an overcomplete Haar-wavelet dictionary and an overcomplete discrete cosine transformation (DCT).

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

ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning. Bruges (Belgium), 28-30 April 2010, d-side publi., ISBN 2-930307-10-2.

2 From vector quantization to sparse coding

A vector quantizer represents each given sample by the closest codebook vector. Vector quantization can be understood as a special case of sparse coding where the coefficients are constrained by $\|\mathbf{a}_i\|_0 = 1$ and $\|\mathbf{a}_i\|_2 = 1$, i.e., vector quantization looks for a codebook C that minimizes (1) choosing the coefficients according to $(\mathbf{a}_i)_m = 1,\ (\mathbf{a}_i)_l = 0\ \forall l \neq m \quad \text{where} \quad m = \arg\min_l \|\mathbf{c}_l - \mathbf{x}_i\|_2^2$. In contrast to hard-competitive learning approaches for vector quantization, the Neural Gas (NG) [4] algorithm considers in each learning step all possible encodings, i.e., $\mathbf{a}_i^1, \ldots, \mathbf{a}_i^M$ with $(\mathbf{a}_i^j)_j = 1$. The encodings are sorted according to their reconstruction error

$$\|\mathbf{x}_i - C\mathbf{a}_i^{j_0}\| \leq \|\mathbf{x}_i - C\mathbf{a}_i^{j_1}\| \leq \cdots \leq \|\mathbf{x}_i - C\mathbf{a}_i^{j_p}\| \leq \cdots \leq \|\mathbf{x}_i - C\mathbf{a}_i^{j_M}\| \ . \quad (2)$$

In each learning iteration every codebook vector $\mathbf{c}_l$ is updated. The update is weighted according to the rank of the encoding that uses the codebook vector $\mathbf{c}_l$. It has been shown in [4] that this type of update is equivalent to a gradient descent on a well-defined cost function.

Here we want to directly apply this ranking approach to sparse coding. 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 $\mathbf{x}_i$

$$\|\mathbf{x}_i - C\mathbf{a}_i^{j_0}\| < \|\mathbf{x}_i - C\mathbf{a}_i^{j_1}\| < \cdots < \|\mathbf{x}_i - C\mathbf{a}_i^{j_p}\| < \cdots < \|\mathbf{x}_i - C\mathbf{a}_i^{j_K}\| \ . \quad (3)$$

Let $\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C) = p$ denote the number of coefficient vectors $\mathbf{a}_i^m$ with $\|\mathbf{x}_i - C\mathbf{a}_i^m\| < \|\mathbf{x}_i - C\mathbf{a}_i^j\|$. Introducing the neighborhood $h_{\lambda_t}(v) = e^{-v/\lambda_t}$, we consider the following modified error function

$$E_s = \sum_{i=1}^{L}\sum_{j=1}^{K} h_{\lambda_t}(\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C))\|\mathbf{x}_i - C\mathbf{a}_i^j\|_2^2 \quad (4)$$

which becomes equal to (1) for $\lambda_t \to 0$. In order to minimize (4), we consider the gradient of E_s with respect to C, which is

$$\frac{\partial E_s}{\partial C} = -2\sum_{i=1}^{L}\sum_{j=1}^{K} h_{\lambda_t}(\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C))(\mathbf{x}_i - C\mathbf{a}_i^j)\mathbf{a}_i^{j^T} + R \quad (5)$$

with

$$R = \sum_{i=1}^{L}\sum_{j=1}^{K} h'_{\lambda_t}(\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C))\frac{\partial \text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C)}{\partial C}\|\mathbf{x}_i - C\mathbf{a}_i^j\|_2^2. \quad (6)$$

Adopting the proof given in [4], it can be shown that $R = 0$. Hence, we can perform a stochastic gradient descent on (4) with respect to C by applying

$t = 0, \ldots, t_{\max}$ updates of C using the gradient based learning rule

$$\Delta C = \alpha_t \sum_{j=1}^{K} h_{\lambda_t}(\mathrm{rank}(\mathbf{x}, \mathbf{a}^j, C))(\mathbf{x} - C\mathbf{a}^j){\mathbf{a}^j}^T \tag{7}$$

for a randomly chosen $\mathbf{x} \in X$ where $\lambda_t = \lambda_0 \left(\lambda_{\mathrm{final}}/\lambda_0\right)^{\frac{t}{t_{\max}}}$ is an exponentially decreasing neighborhood-size and $\alpha_t = \alpha_0 \left(\alpha_{\mathrm{final}}/\alpha_0\right)^{\frac{t}{t_{\max}}}$ an exponentially decreasing learning rate. After each update has been applied the norm of the column vectors of C is set to one. Then a new training sample $\mathbf{x}$ is selected, the corresponding $\mathbf{a}^j$ are determined and the next update of C can be performed.

3 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 k, this approach is not applicable in practice. However, since in (4) 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 such that not only the best but the first K_{user} best coefficients $\mathbf{a}^j$ are determined, at least approximately. 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

$$\mathbf{y}_n^j = \left({\mathbf{r}_1^{n,j}}^T \boldsymbol{\epsilon}_n^j)/\|\mathbf{r}_1^{n,j}\|, \ldots, ({\mathbf{r}_l^{n,j}}^T \boldsymbol{\epsilon}_n^j)/\|\mathbf{r}_l^{n,j}\|, \ldots, ({\mathbf{r}_M^{n,j}}^T \boldsymbol{\epsilon}_n^j)/\|\mathbf{r}_M^{n,j}\|\right) \tag{8}$$

it looks for

$$l_{\mathrm{win}}(n, j) = \arg \max_{l, l \notin U_n^j} (\mathbf{y}_n^j)_l \ . \tag{9}$$

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

$$R_{n+1}^j = R_n^j - \left(\mathbf{r}_{l_{\mathrm{win}}(n,j)}^{n,j} ({R_n^j}^T \mathbf{r}_{l_{\mathrm{win}}(n,j)}^{n,j})^T\right) / ({\mathbf{r}_{l_{\mathrm{win}}(n,j)}^{n,j}}^T \mathbf{r}_{l_{\mathrm{win}}(n,j)}^{n,j}) \ . \tag{10}$$

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

$$\boldsymbol{\epsilon}_{n+1}^j = \boldsymbol{\epsilon}_n^j - \left(({\boldsymbol{\epsilon}_n^j}^T \mathbf{r}_{l_{\mathrm{win}}(n,j)}^{n,j})/({\mathbf{r}_{l_{\mathrm{win}}(n,j)}^{n,j}}^T \mathbf{r}_{l_{\mathrm{win}}(n,j)}^{n,j})\right) \mathbf{r}_{l_{\mathrm{win}}(n,j)}^{n,j} \ . \tag{11}$$

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

$$Q(l,n,j) = \begin{cases} 0: & \text{If there is no pursuit among all pursuits that have} \\ & \text{been performed with respect to } \mathbf{x} \text{ that is equal to} \\ & \text{the } j\text{-th pursuit up to the } n\text{-th iteration where in} \\ & \text{that iteration column } l \text{ has been selected} \\ 1: & \text{else .} \end{cases} \tag{12}$$

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

$$j_{\text{target}} = \arg \max_{j=1,\ldots,m} \max_{n=0,\ldots,s_j-1} \max_{l,Q(l,n,j)=0} (\mathbf{y}_n^j)_l \tag{13}$$

$$n_{\text{target}} = \arg \max_{n=0,\ldots,s_{j_{\text{target}}}-1} \max_{l,Q(l,n,j_{\text{target}})=0} (\mathbf{y}_n^{j_{\text{target}}})_l \tag{14}$$

$$l_{\text{target}} = \arg \max_{l,Q(l,n_{\text{target}},j_{\text{target}})=0} (\mathbf{y}_{n_{\text{target}}}^{j_{\text{target}}})_l \ . \tag{15}$$

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.

4 Experiments

We extracted 10000 random patches of natural images of size 8×8 and learned an overcomplete dictionary based on these patches. Either the K-SVD algorithm or the method proposed in this paper were employed for learning ($\alpha_0 = 0.1, \alpha_{\text{final}} = 0.01, \lambda_0 = 20, \lambda_{\text{final}} = 0.01$). In order to show the gain that is obtained by the soft-competitive learning, we also learned a dictionary where the neighborhood-size for our method was practically zero ($\lambda_0 = \lambda_{\text{final}} = 10^{-10}$), which corresponds to hard-competitive learning. Additionally, we included overcomplete Haar-wavelet and DCT dictionaries in the experiment. The size of the dictionaries was $M = 441$. For k, the number of non-zero coefficients, we used 2, 5 and 10. We performed 10 training epochs, which corresponds to 100000 update steps according to (7). For the K-SVD algorithm, we performed 50 learning iterations, each using 2000 random patches that were randomly chosen every 10 iterations. In all cases the coefficients were obtained from the BOP method that is proposed in Section 3 where the number of pursuits was set to $K_{\text{user}} = 20$. For the K-SVD method only the coefficients of the best pursuit provided by

BOP were used for learning. In order to remove the DC compononent from the image patches, for K-SVD and our method 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 thus implicitly increasing k by one.

From a given image, we removed a certain percentage of all pixels. Then, for each 8×8 patch of the incomplete image we computed the coefficients with respect to a given dictionary using the BOP method ($K_{\text{user}} = 20$). Only the remaining pixels were used, i.e., the dimensions of the dictionary elements corresponding to the missing pixels were not considered. δ, the minimum norm of the residual for the BOP stopping criterion was set to $0.32 = \|0.04 \cdot \mathbf{1}\|$, $\mathbf{1} \in \mathbb{R}^{64}$ which corresponds to an average error of approximately ± 10 in 8-bit grayscale images. In order to reconstruct the image patch, we then took the linear combination of the complete dictionary elements using the best coefficients. Finally, the estimation of each missing pixel at a certain position in the image was obtained as the mean value of all estimated patches at that position.

Table 1 shows the original image, the same image after 70 percent of all pixels have been deleted, as well as the reconstruction that was obtained from the different dictionaries used in the experiments. It can be seen that the learned dictionaries, i.e., K-SVD and our method, clearly outperform the Haar and DCT dictionaries. From Table 2, it can be seen, that the method proposed in this paper leads to the smallest root mean square error and mean absolute error with respect to the original image as well as to the smallest mean number of non-zero coefficients required to approximate the image patches up to accuracy $\delta = 0.32$. This holds for different percentages of missing pixels and different choices for the maximum number of non-zero coefficients.

5 Conclusion

We proposed a generalization of the SCNG approach that outperforms other image encoding dictionaries in image reconstruction experiments in terms of reconstruction accuracy. The new approach can be combined with an arbitrary approximation method for the coefficients of the sparse codes.

References

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

[2] 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.

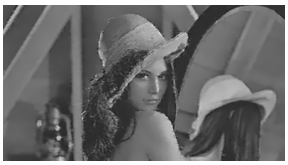
[3] Kai Labusch, Erhardt Barth, and Thomas Martinetz. Sparse Coding Neural Gas: Learning of Overcomplete Data Representations. *Neurocomputing*, 72(7-9):1547–1555, 2009.

[4] 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.

ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning. Bruges (Belgium), 28-30 April 2010, d-side publi., ISBN 2-930307-10-2.

original image

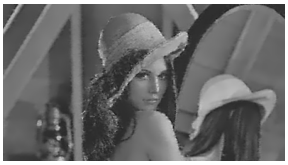
image after 70% of all pixels have been removed



reconstruction obtained from soft competitive learning as proposed in this paper

reconstruction obtained from overcomplete DCT dictionary

reconstruction obtained from overcomplete Haar dictionary



reconstruction obtained from dictionary that was learned using K-SVD

Table 1: Visual comparison of the reconstructions obtained from the different dictionaries. Parameters: $M = 441, k = 5, \delta = 0.32, \alpha_0 = 0.1, \alpha_{\text{final}} = 0.01, \lambda_0 = K_{\text{user}} = 20, \lambda_{\text{final}} = 0.01$.

k, maximum number of nonzero coefficients:	2	2	5	5	10	10
percentage of missing pixels:	50	70	50	70	50	70
RMSE this paper (soft)	**0.0291**	**0.0352**	**0.0260**	**0.0348**	**0.0264**	**0.0352**
RMSE this paper (hard)	0.0299	0.0359	0.0271	0.0358	0.0274	0.0361
RMSE K-SVD	0.0314	0.0379	0.0283	0.0370	0.0290	0.0367
RMSE DCT	0.0341	0.0395	0.0276	0.0377	0.0270	0.0379
RMSE HAAR	0.0381	0.0452	0.0353	0.0436	0.0346	0.0438
MAE this paper (soft)	**0.0161**	**0.0177**	**0.0140**	**0.0169**	**0.0138**	**0.0175**
MAE this paper (hard)	0.0165	0.0182	0.0144	0.0174	0.0142	0.0178
MAE K-SVD	0.0174	0.0194	0.0149	0.0182	0.0149	0.0180
MAE DCT	0.0186	0.0205	0.0148	0.0184	0.0140	0.0186
MAE HAAR	0.0209	0.0233	0.0178	0.0217	0.0170	0.0217
MNZ this paper (soft)	**1.9810**	**2.0899**	**2.8051**	**2.8781**	**3.2520**	**3.0272**
MNZ this paper (hard)	1.9894	2.0967	2.8462	2.9050	3.3012	3.0333
MNZ K-SVD	1.9894	2.0972	2.9064	2.9559	3.5840	3.0741
MNZ DCT	2.0290	2.1338	2.9785	3.0531	3.4791	3.2184
MNZ HAAR	2.0312	2.1346	3.0385	3.0039	3.4949	3.1208

Table 2: Obtained root mean square error (RMSE), mean absolute error (MAE) with respect to the original image and mean number of non-zero coefficients (MNZ) required in order to approximate the incomplete image patches up to an accuracy of δ. Parameters: $M = 441, \delta = 0.32$, $\alpha_0 = 0.1$, $\alpha_{\text{final}} = 0.01$, soft: $\lambda_0 = K_{\text{user}} = 20$, $\lambda_{\text{final}} = 0.01$, hard: $\lambda_0 = \lambda_{\text{final}} = 10^{-10}$, $K_{\text{user}} = 20$.