Controlling Sparseness in Non-negative Tensor Factorization

Matthias Heiler and Christoph Schnörr

Computer Vision, Graphics, and Pattern Recognition Group,
Department of Mathematics and Computer Science,
University of Mannheim, 68131 Mannheim, Germany
{heiler, schnoerr}@uni-mannheim.de

Abstract. Non-negative tensor factorization (NTF) has recently been proposed as sparse and efficient image representation *(Welling and Weber, Patt. Rec. Let., 2001)*. Until now, sparsity of the tensor factorization has been empirically observed in many cases, but there was no systematic way to control it. In this work, we show that a sparsity measure recently proposed for non-negative *matrix* factorization *(Hoyer, J. Mach. Learn. Res., 2004)* applies to NTF and allows precise control over sparseness of the resulting factorization. We devise an algorithm based on sequential conic programming and show improved performance over classical NTF codes on artificial and on real-world data sets.

1 Introduction and Related Work

Non-negative tensor factorization (NTF) has recently been proposed as sparse and efficient image representation [1, 2, 3]. Compared to non-negative *matrix* factorization (NMF) [4, 5], which has also been used for image modeling [6], tensor factorization offers some advantages due to the fact that spacial and temporal correlations are accounted for more accurately than in NMF where images and videos are treated as vectors [7]. In particular, it has been reported that compared to NMF tensor factorization shows a greater degree of sparsity, clearer separation of image parts, better recognition rates, and a tenfold increased compression ratio [3].

From a data analysis viewpoint, NTF is attractive because it usually allows for a *unique decomposition* of a data set into factors. In contrast, while NMF will be unique up to permutation and scaling under some conditions, there are realistic scenarios where additive factors are not separated into independent factors but pollute the whole image basis with "ghost artifacts" [8]. This is not the case with NTF: Under mild conditions, which are usually satisfied by real-world data, tensor factorization is unique [9, 10].

However, until now it was not possible to exercise *explicit sparsity control* with NTF. This differs from NMF where very efficient sparsity control was introduced in [11]. The main problem is that current algorithms for NTF [2] are often variations of general nonlinear programming codes that can be very fast

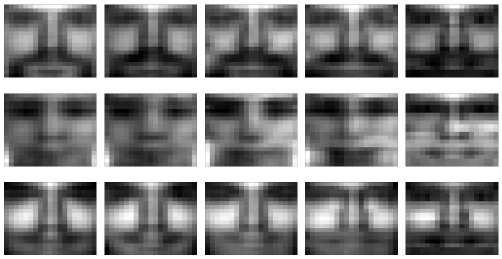


Fig. 1. Sparse NTF face model. MIT CBCL faces are factorized ($k = 10$) and reconstructed using sparsity-control for horizontal factors u_1 (see text). The min-sparsity constraints were 0.0, 0.2, 0.4, 0.6, 0.8 (from left to right). Starting from $s_i^{\min} = 0.4$ reconstructions look increasingly generic and individual features disappear.

as long as sparsity constraints are absent [12, 13]. With additional sparsity constraints the corresponding *projected gradient descent* algorithm [11] can converge slowly. This aggravates with NTF where individual factors interact in a more complicated way.

For sparsity controlled NMF, approaches from convex programming and global optimization [14] have thus been proposed [15]. In this work, we build on such ideas to allow for *fully sparsity-controlled NTF models* and study the behavior of the resulting model on artificial data and on databases of real-world images.

Overview. After this introduction we discuss the sparsity-controlled NTF problem in Sec. 2. In Sec. 3 we provide a practical algorithm to solve the problem. We validate it empirically in Sec. 4 before we summarize our paper in Sec. 5.

Notation. We represent image data as tensor of order 3, e.g., $V \in \mathbb{R}_+^{d_1 \times d_2 \times d_3}$ denotes d_3 images of size $d_1 \times d_2$. We are not concerned about the transformation properties of V, so this simplified 3-way array notation is sufficient. The factorization is given by vectors $u_i^j \in \mathbb{R}^{d_i}$, where $j = 1, \ldots, k$ indexes k independent vectors. Where convenient, we omit indices of the factors, e.g., $u_i \in \mathbb{R}^{d_i \times k}$ is the matrix of k factors corresponding to index i, and u alone is the ordered set of such matrices.

2 The NTF Optimization Problem and Sparseness

In this section we formally state the NTF optimization problem in its original form and extended by sparseness constraints.

2.1 Original NTF Model

The NTF optimization problem admits the general form

$$\begin{aligned} \min_{u_i^j \in \mathbb{R}^{d_i}} \quad & \|V - \sum_{j=1}^{k} \bigotimes_{i=1}^{3} u_i^j \|_F^2 \\ \text{s.t.} \quad & 0 \leq u_i^j . \end{aligned} \tag{1}$$

Here, image volume V is approximated by the sum of k rank-1 tensors that are outer products $u_1^j \otimes u_2^j \otimes u_3^j$. By using outer products with additional factors $u_i^j, i > 3$, this generalizes to higher-order tensors. In this work, however, we are concerned with image volumes only.

It is instructive to compare NTF with the more widespread NMF model: In NMF, image data is first vectorized, and the resulting non-negative matrix $V \in \mathbb{R}_+^{m \times d_3}$, $m = d_1 \cdot d_2$, is then factorized as the product of two non-negative matrices $W \in \mathbb{R}_+^{m \times k}$ and $H \in \mathbb{R}_+^{k \times d_3}$. In short, one optimizes

$$\begin{aligned} \min_{W,H} \quad & \|V - WH\|_F^2 \\ \text{s.t.} \quad & 0 \leq W, H. \end{aligned} \tag{2}$$

It is clear that the vectorized representation does not take into account the spatio-temporal correlations of image data or video. In contrast, the NTF analogon to basis images are rank-one matrices $u_1^j \otimes u_2^j$ that nicely represent correlations along the x and y direction of the image plane. The price to pay is that with NTF basis images are no longer arbitrary: The rank one restriction rules out, e.g., basis images with diagonal structures.

2.2 Sparsity-Constrained NTF

It has early been reported that NTF codes tend do be *sparse*, i.e., many entries of the u_i^j equal zero [1]. Especially for pattern recognition applications, sparsity is a key property since it relates directly to learnability [16, 17] and is biologically well motivated [18]. Sparsity also seems to act as a strong prior for *localized image representations* [11]. Such representations are desirable since they naturally focus on *parts* and thus are potentially *more robust against occlusion or noise* than are their global counterparts.

Thus, the following sparseness measure has been suggested for NMF [11]:

$$\text{sp}(x) := \frac{1}{\sqrt{n} - 1} \left(\sqrt{n} - \frac{\|x\|_1}{\|x\|_2} \right). \tag{3}$$

It assigns to each vector[1] $x \in \mathbb{R}^n \setminus \{0\}$ a real number within $[0, 1]$ where $\text{sp}(x) = 0$ corresponds to a uniform vector with $x_i = const > 0, \forall i$, and $\text{sp}(x) = 1$ corresponds to a vector with a single non-zero element. Since $\text{sp}(x)$ is not affected by multiplicative factors, i.e., $c > 0 \Rightarrow \forall x : \text{sp}(x) = \text{sp}(c \cdot x)$, and varies continuously between the two boundary cases it serves as a convenient and expressive sparsity measure. Empirically, it has been observed that extending (2) by sparseness constraints can lead to considerably improved non-negative basis functions which are more localized and allow easier semantic interpretation [11, 15].

[1] Where convenient, we will also use $\text{sp}(M) \in \mathbb{R}^n$ for matrices $M \in \mathbb{R}^{m \times n}$. Then, sparsity is measured for each column of M and the results are stacked into a vector.

Thus, it is desirable to extend (1) by similar sparsity-controlling constraints, leading to the problem

$$\begin{aligned} \min_{u_i^j \in \mathbb{R}^{d_i}} \quad & \|V - \sum_{j=1}^{k} \bigotimes_{i=1}^{3} u_i^j \|_F^2 \\ \text{s.t.} \quad & 0 \leq u_i \\ & s_i^{\min} \leq \operatorname{sp}(u_i) \leq s_i^{\max}. \end{aligned} \tag{4}$$

The parameters $s_i^{\min}$ and $s_i^{\max}$ are real numbers in $[0, 1]$ specified by the user for a given application. We propose solvers for (4) in Sec. 3 and validate the model on artificial and on real-world data in Sec. 4.

3 Solving Sparsity-Constrained NTF

In this section, we develop an algorithm for solving problem (4). The basic building block of our method are *second order cone programs* (SOCPs) which we introduce in Sec. 3.1. In Sec. 3.2 we propose an algorithm that dually and alternately optimizes sparseness and reconstruction quality of the tensor approximation.

3.1 Sparsity and Second Order Cones

From an optimization viewpoint, it is important to note that (3) models a *second order conic set* [19]. The second order standard cone $\mathcal{L}^{n+1} \subset \mathbb{R}^{n+1}$ is the convex set:

$$\mathcal{L}^{n+1} := \left\{ \begin{pmatrix} x \\ t \end{pmatrix} = (x_1, \ldots, x_n, t)^\top \;\middle|\; \|x\|_2 \leq t \right\}. \tag{5}$$

As second order cones are useful in modeling a range of applications and are computationally convenient at the same time, they gave rise to the framework of second order cone programming [19]. In SOCP one considers problems with conic constraints that admit the general form

$$\begin{aligned} \inf_{x \in \mathbb{R}^n} \quad & f^\top x \\ \text{s.t.} \quad & \begin{pmatrix} A_i x + b_i \\ c_i^\top x + d_i \end{pmatrix} \in \mathcal{L}^{n+1}, \qquad i = 1, \ldots, m. \end{aligned} \tag{6}$$

Being convex problems, efficient and robust solvers for SOCPs exist in software [20, 21, 22]. Furthermore, additional linear constraints and, in particular, the condition $x \in \mathbb{R}_+^n$ are admissible, as they are special cases of constraints of the form (6).

Considering the sparseness function (3) it has been pointed out [15] that the set of non-negative vectors $x \geq 0$ no sparser than $s \in [0, 1]$ is given by the second order cone

$$C(s) := \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{pmatrix} x \\ \frac{1}{c_{n,s}} e^\top x \end{pmatrix} \in \mathcal{L}^{n+1} \right\}, \quad c_{n,s} := \sqrt{n} - (\sqrt{n} - 1)s. \tag{7}$$

In this light, we can rewrite (4) as

$$\begin{aligned} \min_{u_i^j} \quad & \|V - \sum_{j=1}^{k} \bigotimes_{i=1}^{3} u_i^j \|_F^2 \\ \text{s.t.} \quad & u_i^j \in (\mathbb{R}_+^{d_i} \cap C(s_i^{\max})) \setminus C(s_i^{\min}), \quad j = 1, \dots, k. \end{aligned} \tag{8}$$

This notation makes explicit that the constraints consist of a *convex* part $u_i^j \in \{\mathbb{R}_+^{d_i} \cap C(s_i^{\max})\}$ and a *reverse-convex* part $u_i^j \notin C(s_i^{\min})$. The two fundamental challenges to address are thus, first, the non-convex objective function, and, second, the reverse-convex min-sparsity constraint.

3.2 The Sparsity Maximization Algorithm (SMA)

We use two strategies to cope with the basic challenges in problem (8): First, to address the non-convexity of the objective function, we apply an *alternate minimization approach* where only one component u_i, $i \in \{1, 2, 3\}$, is optimized at a time while the other two components are held constant. The resulting objective function is convex quadratic in each step. Alternate minimization is very popular with NMF, NTF, and similar models and seems to perform well in experiments [1, 2, 3, 4, 5, 6, 13, 15]. Note that for problems where memory is not a major concern, *joint* optimizations of pairs or triplets of the u_i components may offer performance benefits, especially toward the end of an optimization [5, 23]. For our sparsity maximization-approach, however, we will remain with the more memory efficient and simpler scheme of strict alternate minimization.

To deal with the second challenge, the reverse-convex min-sparsity constraint, we adopt an approach from global optimization [24]: Given a current estimate for u_i we compute the maximally sparse approximation subject to the constraint that the reconstruction error does not deteriorate, and, dually, given a maximally sparse approximation we minimize the reconstruction error subject to the constraint that the min-sparsity constraint may not be violated.

Let us assume that within the alternate minimization approach ("outer loop") we optimize component u_i, while the components $\bar{I} := \{1, 2, 3\} \setminus \{i\}$ remain fixed. Then the target function $f(V, u) := \|V - \sum_{j=1}^{k} \bigotimes_{i=1}^{3} u_i^j \|_F^2$ can be written as $f(V, u_i) = \|\text{vec}(V) - U\text{vec}(u_i)\|_2^2$, where U is a sparse matrix containing the corresponding entries u_i, $i \in \bar{I}$, that are not currently optimized.

Initialization. We start with any u_i that obeys the constraints of (8). A simple way to obtain such an initialization is to first solve the problem ignoring the min-sparsity constraint, i.e.,

$$\begin{aligned} \min_{u_i} \quad & f(V, u_i) \\ \text{s.t.} \quad & u_i^j \in \mathbb{R}_+^{d_i} \cap C(s_i^{\max}), \quad j = 1, \dots, k \end{aligned} \tag{9}$$

which is a SOCP that reads in standard form

$$\begin{aligned} \min_{u_i, z} \quad & z \\ \text{s.t.} \quad & 0 \le u_i \\ & \begin{pmatrix} \operatorname{vec}(V) - U \operatorname{vec}(u_i) \\ z \end{pmatrix} \in \mathcal{L}^{kd_i+1} \\ & \begin{pmatrix} u_i^j \\ (c_{d_i, s_i^{\max}})^{-1} e^\top u_i^j \end{pmatrix} \in \mathcal{L}^{d_i+1}, \quad j = 1, \dots, k. \end{aligned} \tag{10}$$

The resulting u_i can then be projected on the boundary of the min-sparsity cone. Accuracy is of no concern in this step, so simple element-wise exponentiation followed by normalization

$$\pi(u_i^j) \propto \frac{(u_i^j)^t}{\|(u_i^j)^t\|_2} \tag{11}$$

with suitable parameter t, yields a feasible initialization.

Step one. In the first step we maximize worst-case sparsity subject to the constraint that reconstruction accuracy may not deteriorate:

$$\begin{aligned} \max_{u_i} \quad & \min_j \operatorname{sp}(u_i^j) \\ \text{s.t.} \quad & u_i^j \in \mathbb{R}_+^{d_i} \cap C(s_i^{\max}), \quad j = 1, \dots, k \\ & f(V, u_i) \le f(V, \bar{u}_i), \end{aligned} \tag{12}$$

where $\bar{u}_i$ is the estimate for u_i before sparsity maximization. Problems similar to (12) have been solved using cutting plane methods, however, such solvers seem to perform well for small to medium-sized problems only [24, 14]. For the large scale problems common in computer vision and machine learning, we must content ourselves with a local solution obtained by *linearization* of the sparsity cone around the current estimate $\bar{u}_i$. The resulting problem is a SOCP:

$$\max_{u_i, z} \quad z \tag{13a}$$

$$\text{s.t.} \quad u_i^j \in \mathbb{R}_+^{d_i} \cap C(s_i^{\max}), \quad j = 1, \dots, k \tag{13b}$$

$$f(V, u_i) \le f(V, \bar{u}_i) \tag{13c}$$

$$z \le \operatorname{sp}(\bar{u}_i^j) + \langle \nabla \operatorname{sp}(\bar{u}_i^j), u_i^j - \bar{u}_i^j \rangle, \quad j = 1, \dots, k. \tag{13d}$$

Note that $\operatorname{sp}(x)$ is convex, so the linearization (13) is valid in the sense that min-sparsity will never decrease in step one.

Step two. In the second step we improve the objective function while paying attention not to violate the min-sparsity constraints. Given the sparsity-maximized estimate $\bar{u}_i$ we solve the SOCP

$$\min_{u_i} \quad f(V, u_i) \tag{14a}$$

$$\text{s.t.} \quad u_i^j \in \mathbb{R}_+^{d_i} \cap C(s_i^{\max}), \quad j = 1, \dots, k \tag{14b}$$

$$\|u_i^j - \bar{u}_i^j\|_2 \le \min_{q \in C(s_i^{\min})} \|q - \bar{u}_i^j\|_2, \quad j = 1, \dots, k \tag{14c}$$

Algorithm 1. The sparsity maximization algorithm in pseudocode

```
 1: initialize all $u_i^j$ using eqn. (10) and (11), set $\bar{u} \leftarrow u$
 2: repeat
 3:   for $i = 1$ to 3 do
 4:     repeat
 5:       $u_{\text{old}} \leftarrow u$
 6:       $\bar{u}_i \leftarrow$ solution of (13)
 7:       $u_i \leftarrow$ solution of (14)
 8:     until $|f(V, u_i) - f(V, u_{\text{old},i})| \leq \epsilon$
 9:   end for
10: until no improvement found in loop 3–9
```

which is straightforward to translate to standard form. Note that constraints (14c) make sure that the resulting u_i^j will not enter the min-sparsity cone. In effect, the reverse-convex min-sparsity constraint is translated in (14) into a convex proximity constraint. This is similar to *trust region* approaches common in nonlinear programming.

Termination. After the second step we check whether $f(V, u_i)$ improved more than ϵ. If it did we jump to step one, otherwise we switch in the outer loop to a different factor i. The whole algorithm is outlined in Alg. 1.

3.3 Convergence Properties

Regarding termination of Alg. 1, we assert:

Proposition 1. *The SMA algorithm (Alg. 1) terminates in finite time for any sparsity-constrained NTF problem.*

Proof (sketch). For lack of space, we omit technicalities and note that:

- Step 1 consists of solving three convex programs and subsequent projections. These operations will terminate in polynomial time.
- Any current estimate u is a feasible point for the convex programs (polynomial time) in the inner loop (steps 6 and 7). Thus, with each iteration of the inner loop the objective value $f(V, u)$ can only decrease or remain constant.
- Since $f(V, u)$ is bounded from below, the inner loop will eventually terminate (step 8).
- And so will the outer loop (step 10) for the same reason. □

The algorithm conveniently converges on a stationary point if the constraints are *regular*. Following [24] we call constraints regular if their gradients are linearly independent and if removing one would allow for a new optimum with lower objective value. From a practical viewpoint, this means that in particular we assume $s_i^{\min} < s_i^{\max}$, i.e., the interior of the feasible set is not empty.

Proposition 2. *Under regular sparsity constraints, Alg. 1 converges on a stationary point of problem* (4)*.*

Proof. The first order optimality conditions for problem (4) read:

$$
\begin{aligned}
-\frac{\partial L}{\partial u_i^*} &\in N_{Q_i}(u_i^*), &\text{(15a)}\\
G_i(u_i^*) &\in \mathbb{R}_+^k, &\text{(15b)}\\
\lambda_i^* &\in R_-^k, &\text{(15c)}\\
\langle \lambda_i^*, u_i^* \rangle &= 0, &\text{(15d)}
\end{aligned}
$$

where i runs from 1 to 3. Here,

$$L(u, \lambda_1, \lambda_2, \lambda_3) = f(V, u) + \sum_{i=1}^{3} \lambda_i^\top G_i(u_i) \tag{16}$$

is the Lagrangean of the problem and

$$G_i(u_i) = \Big(\|u_i^1\|_2 - (c_{d_i, s_i^{\min}})^{-1} \|u_i^1\|_1, \cdots, \|u_i^k\|_2 - (c_{d_i, s_i^{\min}})^{-1} \|u_i^k\|_1 \Big)^\top \tag{17}$$

encodes the min-sparsity constraints: $G_i(u_i)$ is non-negative if the min-sparsity constraints on u_i are adhered to. Finally, N_{Q_i} in (15a) is the normal cone [25] to the convex set $R_+^{d_i \times k} \cap C(s_i^{\max}),\ i = 1, \ldots, 3$.

Now assume the algorithm converged (Prop. 1) on a point $\tilde{u}$. Because $\mathrm{sp}(\cdot)$ is convex and the constraints are regular we find that (13d) is locally equivalent to $z \le \mathrm{sp}(\tilde{u}_i)$. In fact, $z = s_i^{\min}$ because the min-sparsity constraint is active for some vector $\tilde{u}_i^j$: Otherwise we could remove the constraint without changing the objective value of the solution.

Overall, we find that the solution to (13) satisfies

$$
\begin{aligned}
\max_{z, u_i \in Q_i} \quad & z, \\
\text{s.t.} \quad & z = \min_i \mathrm{sp}(u_i), \\
& 0 \le f(V, u_i) - f(V, \tilde{u}_i), \\
& G_i(u_i) \in \mathbb{R}_+^k.
\end{aligned} \tag{18}
$$

Then the solution obeys the corresponding first order condition

$$-\frac{\partial}{\partial u_i} \Big(\hat{\lambda}_{fi} f(V, u_i) + \langle \hat{\lambda}_{ui}, G_i(u_i) \rangle \Big) \in N_{Q_i}(u_i^*) \tag{19}$$

which is equivalent to (15). □

3.4 Practical Considerations

The SOCP problems (13) and (14) are sparse but can become very large. Solvers with support for sparse matrices are crucial[2]. In applications where the convex max-sparsity constraints are not used, i.e., only min-sparsity constraints are specified, quadratic programming (QP) solvers can be used instead of SOCP solvers. Commercial QP solvers are usually highly optimized and may be faster than their SOCP counterparts.

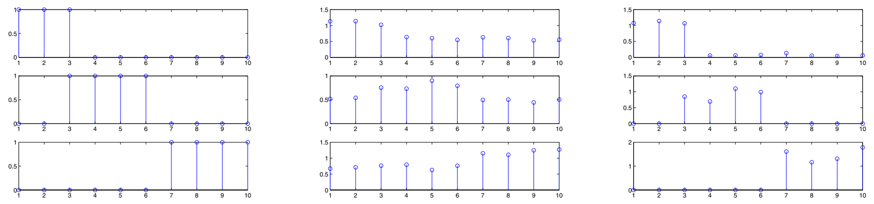
[2] In our experiments we used MOSEK 3.2.1.8 [22].

4 Experiments

In this section we show that our optimization framework works robustly in practice. A comparison demonstrates that explicit sparsity-control leads to improved performance. Our results validate that sparsity-controlled NTF can be a useful model in real applications.

4.1 Ground Truth Experiment

To validate our approach we created an artificial data set with known ground truth. Specifically, we used three equally-sized factors u_i with $d_i = 10$ and all entries zero except for the entries shown in Fig. 2(a). We computed $V = u_1 \otimes u_2 \otimes u_3 + |\nu|$, where $\nu \sim \mathcal{N}(0, 0.5)$ was i.i.d. Gaussian noise.



(a) Ground truth (b) Classical NTF (c) Sparse NTF

Fig. 2. Ground truth experiment. We created an artificial data set with known factors u_i (Fig. 2(a)). We added noise (see text) and used NTF to recover the factors from $V = u_1 \otimes u_2 \otimes u_3 + |\nu|$. While the NTF model without sparsity constraints failed (Fig. 2(b)), sparsity-controlled NTF successfully recovered the factors (Fig. 2(c)).

We found that over 10 repeated runs the classical NTF model without sparsity constraints was not able to recover any of the factors (Fig. 2(b)). In contrast, sparsity-controlled NTF with $s_i^{\min} = 0.55$ yielded useful results in all 10 repeated runs (Fig. 2(c)).

We conclude that in the presence of noise, sparsity constraints are crucial to successfully recover sparse factors. Further, we find that at least with the simple data set above the sparsity maximization algorithm converged on the correct factorization in 10 out of 10 repeated runs.

4.2 Face Detection

For the face detection problem, impressive results are reported in [3] where NTF without sparsity constraints clearly outperformed NMF recognition rates on the MIT CBCL face data set [26]. We demonstrate in this section that performance can further be improved by using sparsity-constrained NTF.

In our experiments we used the original training and test data sets provided by CBCL [26]. In this data sets, especially the test data set is very imbalanced:

Table 1. Recognition performance of sparse NTF codes. We trained a SVM on a subset of the MIT CBCL face detection data set (see text). Features were raw pixels, a NMF basis, and a NTF basis with different min-sparsity constraints. We compared area under ROC for the MIT training data (first row), the MIT test data set (second row) and recognition accuracy for a balanced test data set with 50% face samples (last row). NTF with a relatively strong min-sparsity constraint $s_1^{\min} = 0.8$ performs best.

feature $s_1^{\min}$	pixels	NMF	NTF 0.0	NTF 0.3	NTF 0.4	NTF 0.6	NTF 0.7	**NTF** **0.8**	NTF 0.9
ROC (trai)	0.997	0.995	1.000	1.000	0.997	0.997	0.994	**1.000**	0.991
ROC (test)	0.817	0.817	0.835	0.822	0.789	0.830	0.822	**0.860**	0.821
ACC-50 (test)	0.611	0.667	0.753	0.600	0.702	0.743	0.728	**0.761**	0.719

A trivial classificator returning “non-face” for all input would obtain 98% accuracy. For this reason, we consider the *area under the ROC curve* as a more suitable performance measure. We thus trained radial-basis function SVMs on small subsets (250 samples only) of the CBCL training data set. To determine the SVM and kernel parameters, we used 5-fold crossvalidation on the training data. For the resulting SVM we determined the area under the ROC on the test data set. In addition, we also created a data set ACC-50 consisting of all 472 positive samples in the test data set as well as of 472 randomly chosen negative test samples.

We compared the following feature sets:

1. the $19 \times 19 = 361$ raw image pixels as found in the CBCL data set,
2. coefficients for 10 NMF basis functions determined on a subset of the faces in the training data set,
3. coefficients for 10 NTF basis functions determined on a subset of the faces in the training data set using different values of the min-sparsity constraint on u_1. Reconstructions using these features are shown in Fig. 1. Note that the NTF basis corresponds to an about 10-fold higher compression ratio than the NMF basis.

The results are summarized in Tab. 1: NMF and raw pixel values perform similar in this experiment. NTF yields improved results, which is consistent with [3]. Best results are obtained with NTF with strong sparsity constraint ($s_1^{\min} = 0.8$).

5 Conclusions

We extended the non-negative tensor factorization model for images [1, 2, 3] by explicit sparseness constraint [11]. We found that compared to unconstrained NTF the extended model can be more robust against noise (Sec. 4.1) and the corresponding image codes can be more efficient for recognition, especially when training data is scarce (Sec. 4.2).

From an optimization point of view, we devised an algorithm based on sequential conic programming (Sec. 3.2) which has desirable convergence properties (Sec. 3.3) and works well in practice (Sec. 4). Because the algorithm's basic building blocks are convex programs, we believe the model could further be extended by additional convex constraints taking into account prior knowledge about the specific problem at hand, while still remaining in the sequential convex programming framework.

References

1. M. Welling and M. Weber, “Positive tensor factorization,” *Pattern Recog. Letters*, vol. 22, no. 12, pp. 1255–1261, 2001.
2. A. Shashua and T. Hazan, “Non-negative tensor factorization with applications to statistics and computer vision,” in *Proc. of ICML*, 2005.
3. T. Hazan, S. Polak, and A. Shashua, “Sparse image coding using a 3D non-negative tensor factorization,” in *Proc. of ICCV*, 2005.
4. J. Shen and G. W. Israël, “A receptor model using a specific non-negative transformation technique for ambient aerosol,” *Atmospheric Environment*, vol. 23, no. 10, pp. 2289–2298, 1989.
5. P. Paatero and U. Tapper, “Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values,” *Environmetrics*, vol. 5, pp. 111–126, 1994.
6. D. D. Lee and H. S. Seung, “Learning the parts of objects by non-negative matrix factorization,” *Nature*, vol. 401, pp. 788–791, Oct. 1999.
7. A. Shashua and A. Levin, “Linear image coding for regression and classification using the tensor-rank principle,” in *Proc. of CVPR*, 2001.
8. D. Donoho and V. Stodden, “When does non-negative matrix factorization give a correct decomposition into parts?,” in *Adv. in NIPS*, vol. 17, 2004.
9. J. B. Kruskal, “Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics,” *Linear Algebra and its Applications*, vol. 18, pp. 95–138, 1977.
10. N. D. Sidiropoulos and R. Bro, “On the uniqueness of multilinear decompositions of N-way arrays,” *J. of Chemometrics*, vol. 14, pp. 229–239, 2000.
11. P. O. Hoyer, “Non-negative matrix factorization with sparseness constraints,” *J. of Mach. Learning Res.*, vol. 5, pp. 1457–1469, 2004.
12. D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” in *Adv. in NIPS*, 2000.
13. M. Chu, F. Diele, R. Plemmons, and S. Ragni, “Optimality, computations, and interpretation of nonnegative matrix factorizations.” submitted to SIAM J. Mat. Anal. Appl., 2004. www4.ncsu.edu:8030/∼mtchu/Research/Papers/nnmf.pdf.
14. R. Horst and H. Tuy, *Global Optimization.* Springer, Berlin, 1996.
15. M. Heiler and C. Schnörr, “Learning non-negative sparse image codes by convex programming,” in *Proc. of ICCV*, 2005.
16. N. Littlestone and M. Warmuth, “Relating data compression, learnability, and the Vapnik-Chervonenkis dimension,” tech. rep., Univ. of Calif. Santa Cruz, 1986.
17. R. Herbrich and R. C. Williamson, “Algorithmic luckiness,” *J. of Mach. Learning Res.*, vol. 3, pp. 175–212, 2002.
18. B. A. Olshausen and D. J. Field, “Sparse coding with an overcomplete basis set: A strategy employed by V1?,” *Vision Research*, vol. 37, pp. 3311–3325, Dec. 1997.
19. M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, “Applications of second-order cone programming,” *Linear Algebra and its Applications*, 1998.

20. J. F. Sturm, *Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones (updated version 1.05).* Department of Econometrics, Tilburg University, Tilburg, The Netherlands, 2001.
21. H. Mittelmann, “An independent benchmarking of SDP and SOCP solvers.,” *Math. Programming, Series B*, vol. 95, no. 2, pp. 407–430, 2003.
22. MOSEK ApS, Denmark, *The MOSEK optimization tools version 3.2 (Revision 8) User’s manual and reference*, 2005.
23. A. M. Buchanan and A. W. Fitzgibbon, “Damped Newton algorithms for matrix factorization with missing data,” in *CVPR05*, vol. 2, pp. 316–322, 2005.
24. H. Tuy, “Convex programs with an additional reverse convex constraint,” *J. of Optim. Theory and Applic.*, vol. 52, pp. 463–486, Mar. 1987.
25. R. Rockafellar and R.-B. Wets, *Variational Analysis*, vol. 317 of *Grundlehren der math. Wissenschaften.* Springer, 1998.
26. CBCL, “CBCL face database #1.” MIT Center For Biological and Computational Learning, http://cbcl.mit.edu/software-datasets, 2000.