Sparse coding and NMF

Julian Eggert, Edgar Körner

2004

Preprint:

This is an accepted article published in 2004 IEEE International Joint Conference on Neural Networks. The final authenticated version is available online at: https://doi.org/[DOI not available]

Proc. of the IEEE International Joint Conference on Neural Networks IJCNN 2004, pp. 2529-2533

Sparse coding and NMF

Julian Eggert and Edgar Körner
HONDA Research Institute Europe GmbH
Carl-Legien-Straße 30
63073 Offenbach/Main, Germany
E-mail: {Julian.Eggert,Edgar.Koerner}@honda-ri.de

***Abstract*— Non-negative matrix factorization (NMF) is a very efficient parameter-free method for decomposing multivariate data into strictly positive activations and basis vectors. However, the method is not suited for overcomplete representations, where usually sparse coding paradigms apply. In this paper, we show how to merge the concepts of non-negative factorization with sparsity conditions. The result is a multiplicative algorithm that is comparable in efficiency to standard NMF, but that can be used to gain sensible solutions in the overcomplete cases. This is of interest e.g. for the case of learning and modeling of arrays of receptive fields arranged in a visual processing map, where an overcomplete representation is unavoidable.**

I. Introduction

NMF has been introduced by Lee and Seung [3], [4] as an effective factorization method for decomposing multivariate data under the constraints of non-negative components. These constraints result in a representation that is parts-based, because the framework allows only additive, and not substractive, combinations of components. It has been shown [3] that the non-negativity property leads to very different representations than other factorization techniques such as principal components analysis and vector quantization, which learn more holistic representations.

Whereas the non-negativity constraint by itself already enforces some sparseness on the resulting representation of data when using NMF, there is no quantitative control over the "degree" of sparseness included in the algorithm. So far, NMF has been mainly used to generate compressed representations of the data, and in these cases reasonable results are obtained. Nevertheless, there are cases in which an additionally enforced sparseness is required. This is e.g. the case when an overcomplete representation in terms of activities and basis vectors is explored. (Overcomplete meaning that the dimensionality of the space spanned by the factorization code is larger than the effective dimensionality of the input space.) Such overcomplete representations have been used in combination with the requirement of sparse activations to learn and reproduce receptive field profiles of the mammal visual system with great success (see e.g. [5]). To apply NMF to these and similar domains of research which require overcomplete representations, it is therefore desirable to include sparsity constraints explicitly into the advantageous algorithmic framework of NMF.

The remaining sections of the paper are organized as follows. First, in section II, we explore how to incorporate sparsity terms for the activity into the NMF formulation. In section III, we extend on the problem of basis vector normalization that arises when including sparsity into the NMF framework. We formulate the modified NMF algorithm, this time including sparseness constraints, and show how to implement it in an effective way. Finally, in section IV, we present simulation results gained using the modified algorithm.

II. Sparseness and the NMF framework

A. Euclidean NMF algorithm

In the NMF formulation, the following formal problem is considered. We start with a non-negative matrix V containing the original input data and the task is to find non-negative matrices W and H such that their linear combination (the reconstruction of V) R is close to V,

$$V \approx R(W,H) = WH \ . \tag{1}$$

Eq. (1) can be rewritten on a column by column basis as

$$\mathbf{V}_i \approx \mathbf{R}_i(W,H) = \sum_j H_i^j \mathbf{W}_j \ , \tag{2}$$

with $\mathbf{V}_i = V_i^l$, $\mathbf{R}_i = R_i^l$ and $\mathbf{W}_j = W_j^l$ beeing the columns of V, R and W, respectively, and H_i^j the activities used for the encoding of the data (in the following sections, we will omit the (W,H)-dependency for the reconstruction R). Here, the index i in $\mathbf{V}_i$ and $\mathbf{R}_i$ accounts for the i'th input and reconstruction vectors, whereas the index j accounts for the j'th basis vector $\mathbf{W}_j$ used for the reconstruction. This means that each data vector $\mathbf{V}_i$ is approximated by its reconstruction $\mathbf{R}_i$ which is a linear combination of the basis vectors $\mathbf{W}_j$ weighted by their activities H_i^j.

In the following we use an Euclidean cost function that quantifies the degree to which this approximative reconstruction has been achieved by

$$F(W,H) = \frac{1}{2}||V - WH||^2 \ , \tag{3}$$

or, equivalently,

$$F(W,H) = \frac{1}{2}\sum_i \left|\left| \mathbf{V}_i - \sum_j H_i^j \mathbf{W}_j \right|\right|^2 \ . \tag{4}$$

It is shown in [4] that the following NMF update rule(s) minimize (4) with respect to W and H, while preserving non-negativity in all its components. First, the matrices W and

H are initialized with components that are non-negative and larger than zero. Then, we proceed according to:

1) Calculate the reconstruction according to

$$\mathbf{R}_i = \sum_j H_i^j \mathbf{W}_j \ . \tag{5}$$

2) Update the activities according to [1]

$$H_i^j \leftarrow H_i^j \odot \frac{\mathbf{V}_i^T \mathbf{W}_j}{\mathbf{R}_i^T \mathbf{W}_j} \ . \tag{6}$$

3) Calculate the reconstruction with the new activities according to

$$\mathbf{R}_i = \sum_j H_i^j \mathbf{W}_j \ . \tag{7}$$

4) Update the basis vectors according to

$$\mathbf{W}_j \leftarrow \mathbf{W}_j \odot \frac{\sum_i H_i^j \mathbf{V}_i}{\sum_i H_i^j \mathbf{R}_i} \ . \tag{8}$$

5) Return to point 1 until convergence.

(Note that the multiplications and divisions in (6) and (8) are applied componentwise, this is indicated with the $\odot$ for the multiplications.) These update rules are eventually combined with a normalization of the basis vectors to prevent joint numerical drifts in W and H, according to

$$\mathbf{W}_j \leftarrow \frac{\mathbf{W}_j}{\alpha_j} \ , \tag{9}$$

with some norm $\alpha_j := \alpha(\mathbf{W}_j)$. Care has to be taken that the cost function (4) is not changed by the normalization procedure; this is achieved if we simultaneously rescale the activities H_i^j with $1/\alpha_j$. Luckily, the multiplicative update rule for the activities (6) does this automatically but we mention this here because in the sparse framework this is not straightforward.

The update rules can be understood in a gradient descent framework (meaning an additive update rule which modifies each component in the direction of the negative gradient of the cost function (4), proportional to some stepsize variable), with diagonally rescaled stepsize variables, which causes the stepsize parameters to disappear (see eqs. 6 and 7 in [4] for a hint on this issue) and which results in the purely multiplicative update rules (6) and (8).

An advantage of NMF over standard gradient descent updating is that convergence is guaranteed and no stepsize parameter is needed [2]. This means that NMF is a parameter-free gradient descent method, furthermore, the relaxation towards a local minimum is fast and computationally inexpensive.

[1]The multiplication and division are componentwise, so that it is $(\mathbf{v} \odot \mathbf{w})^i := v^i w^i$.

[2]In a restricted sense, NMF automatically chooses an optimal stepsize.

B. Including sparseness for the activity

As motivated in the introduction, for some cases it is sensible to find factorizing solutions W and H to the problem (1) which explicitly enforce a sparseness in their components, in particular in their activations H_i^j. One way to enforce sparseness is by including it explicitly into the cost function

$$F(W,H) = \frac{1}{2} \sum_i \left|\left| \mathbf{V}_i - \sum_j H_i^j \mathbf{W}_j \right|\right|^2 + \lambda \sum_{i,j} g(H_i^j) \ , \tag{10}$$

with the sparsity parameter $\lambda \geq 0$ and the sparsity function g. We see that F is a linear composition of a reconstruction term and a sparsity penalty term.

With eq. (10) we have a scaling problem, since we may scale the basis vectors $\mathbf{W}_j$ with constants α_j and correspondingly the activities H_i^j with constants $1/\alpha_j$ and obtain the same reconstruction cost, but different sparsity penalty. This means that we can always scale up the basis vectors and scale down the activities to get a lower cost function; basically leading to the optimal solution when the sparsity term approaches zero and the basis vectors grow without bounds. Algorithmically, this expresses into a constant shift towards lower activities during the minimization of (10); so that the solutions found do ultimately not depend on the sparsity term any more.

III. Sparse coding in detail

A. Cost function with normalized basis vectors

To avoid the scaling misbehavior of (10), usually a normalization step for the basis vectors such as in (9) is incorporated when minimizing (10). The problem is that a normalization such as in (9) now does not work any more, since it may increase the new, sparse cost function (10) [3]. We therefore aim for a solution for which the normalization constraints are stated in the appropriate way right from the starting point, so that we reformulate the cost function to work with normalized vectors, in the sense that

$$F(W,H) = \frac{1}{2} \sum_i \left|\left| \mathbf{V}_i - \sum_j H_i^j \frac{\mathbf{W}_j}{||\mathbf{W}_j||} \right|\right|^2 + \lambda \sum_{i,j} g(H_i^j) \ , \tag{11}$$

Now we are looking at a cost function which effectively depends on variables $F(\{\bar{\mathbf{W}}_j\}_j, H)$, with $\bar{\mathbf{W}}_j$ being the *normalized* basis vectors,

$$\bar{\mathbf{W}}_j := \frac{\mathbf{W}_j}{||\mathbf{W}_j||} \tag{12}$$

[3]For standard gradient descent, with fixed and identical stepsize parameter for all basis vector variables, a straightforward normalization of the basis vectors is indeed allowed, since even after the normalization it can be shown that the projection on the true gradient remains greater or equal than zero, resulting in a decrease of the cost function and essentially leading to the same local extrema, this has been exploited previously in numerous works. Nevertheless, for NMF, it cannot be guaranteed that the normalization always causes the cost function to decrease. This can be explained if we understand that using NMF, the stepsize for the individual $\mathbf{W}_j$'s is each scaled differently. In effect, it can be seen in simulations that no sensible solutions are found if we simply introduce a basis vector normalization step into the sparse NMF formulation from (10).

(here, $||\dots||$ stands for any differentiable norm). We are then searching for the normalized basis vectors $\bar{\mathbf{W}}_j$ and the activities H_i^j that minimize (11).

Isn't eq. (11) then simply the same as eq. (10) rewritten with $\bar{\mathbf{W}}_j$ instead of $\mathbf{W}_j$? Not quite, since we have the freedom to search the minimum in terms of the original (i.e., non-normalized) basis vectors $\mathbf{W}_j$, and we can renormalize the $\mathbf{W}_j$'s after each step without danger because the value of the cost function remains the same even after applying the normalization. We apply the minimum search on $F(W, H)$ (or, written differently $F(\{\mathbf{W}_j\}_j, H)$), and not on $F(\{\bar{\mathbf{W}}_j\}_j, H)$. This allows us to circumvent the normalization problem of the basis vectors.

B. Modified update rules: NMF with sparseness constraints

The standard NMF update rules can be understood as being gained from the original cost function (4), using gradient descent, separating positive and negative terms, and observing that the fixed point of the update rules is reached when the gradient approaches zero, respectively the quotient approaches 1 (of course this is only a shortcut to the NMF approach, since it does not consider convergence). Similarly to the quotients of the standard NMF, we can reformulate the fixed point conditions for the cost function (11) into multiplicative update rules.

We then get that the complete algorithm for sparse NMF is given by the following set of rules:

1) Calculate and store $\nabla_W ||\mathbf{W}_j||$, for the case that they are needed later (in point 6).
2) Normalize the basis vectors according to $\mathbf{W}_j \leftarrow \mathbf{W}_j / ||\mathbf{W}_j|| = \bar{\mathbf{W}}_j$. This does no harm to the calculation since the cost function does not depend on the norm of the basis vectors.
3) Calculate the reconstruction according to
$$\mathbf{R}_i = \sum_j H_i^j \bar{\mathbf{W}}_j \ . \tag{13}$$
4) Update the activities according to
$$H_i^j \leftarrow H_i^j \odot \frac{\mathbf{V}_i^T \bar{\mathbf{W}}_j}{\mathbf{R}_i^T \bar{\mathbf{W}}_j + \lambda g'(H_i^j)} \ . \tag{14}$$
5) Calculate the reconstruction with the new activities according to
$$\mathbf{R}_i = \sum_j H_i^j \bar{\mathbf{W}}_j \ . \tag{15}$$
6) Update the non-normalized basis vectors according to
$$\mathbf{W}_j \leftarrow \mathbf{W}_j \odot \frac{\sum_i H_i^j \left[\mathbf{V}_i + \left(\mathbf{R}_i^T \bar{\mathbf{W}}_j\right) \nabla_W ||\mathbf{W}_j||\right]}{\sum_i H_i^j \left[\mathbf{R}_i + \left(\mathbf{V}_i^T \bar{\mathbf{W}}_j\right) \nabla_W ||\mathbf{W}_j||\right]} \ . \tag{16}$$
Here use the previously calculated $\nabla_W ||\mathbf{W}_j||$ from point 1.
7) Return to point 1 until convergence.

Two simplifications can be made. We often take a simple linear sparsity function $g(x) = x$, then the activity update rule (14) reduces to
$$H_i^j \leftarrow H_i^j \odot \frac{\mathbf{V}_i^T \bar{\mathbf{W}}_j}{\left[\mathbf{R}_i^T \bar{\mathbf{W}}_j + \lambda\right]} \ . \tag{17}$$
In addition, in case of an Euclidean norm for the basis vectors the notation gets more compact because then the basis vector update rule (16) from point 6 is given by
$$\mathbf{W}_j \leftarrow \mathbf{W}_j \odot \frac{\sum_i H_i^j \left[\mathbf{V}_i + \left(\mathbf{R}_i^T \bar{\mathbf{W}}_j\right) \bar{\mathbf{W}}_j\right]}{\sum_i H_i^j \left[\mathbf{R}_i + \left(\mathbf{V}_i^T \bar{\mathbf{W}}_j\right) \bar{\mathbf{W}}_j\right]} \ , \tag{18}$$
and point 1 can be skipped (from point 7 the algorithm then returns directly to the normalization of the basis vectors, point 2).

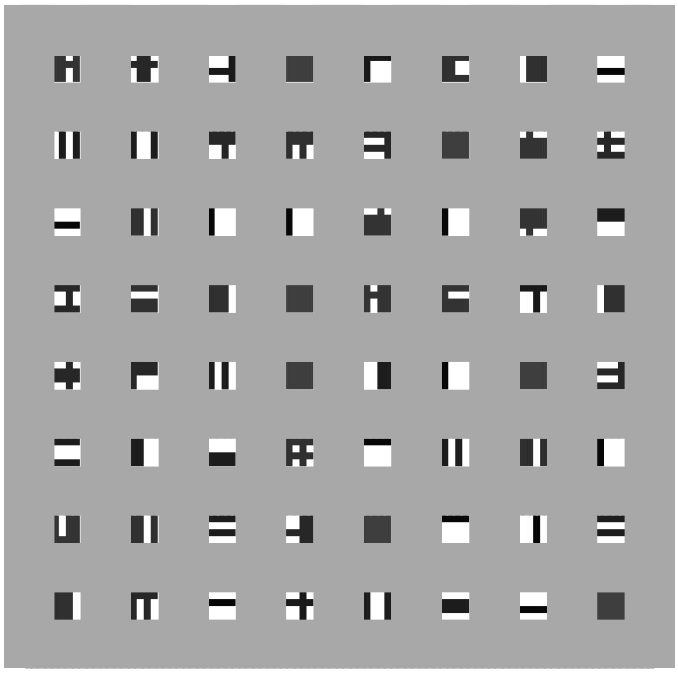


Fig. 1. An extraction of 64 of the 4x4-pixel-sized input images used as input for learning the basis vectors. They are generated by linear superposition of 1 to 4 randomly selected horizontal or vertical lines at arbitrary positions. An additional threshold reduces all pixels to values 0 and 1, afterwards the images have been normalized using an Euclidean norm. Darker colors quantify high values (black is 1) and lighter colors low values (white is 0).

IV. Results and Discussion

A. Simulation results

In this section we show how sparse NMF performs with overcomplete representations. We will follow a somewhat similar line of argumentation as presented in [2]. For simplicity, we use 4×4 pixel images as data vectors. Figure 1 shows a few of the input images used for the task. They are generated by superposition of horizontal and vertical lines, thresholding and normalization. The task is to find the set of basis vectors that properly encode the input images.

Using 8 allowed basis vectors, the standard NMF algorithm finds the correct solution, which consists of all horizontal and vertical lines at the $(4 + 4)$ different positions. This is shown

in fig. 2. The inclusion of a small sparsity term does not make any difference here, i.e., the solutions found by sparse NMF are qualitatively equivalent.

Fig. 2. With 8 allowed basis vectors, the NMF correctly finds the solution of 4 horizontal and 4 vertical lines (1 at each position).

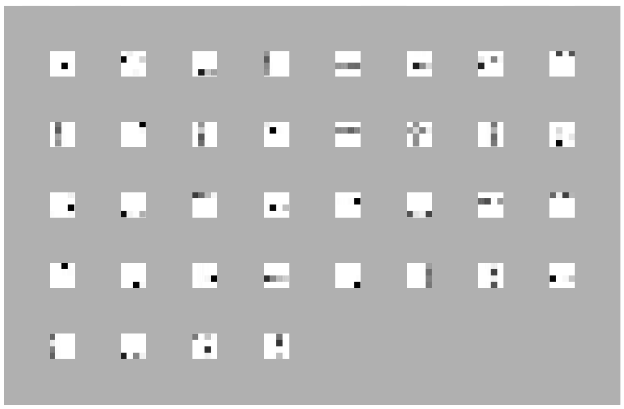
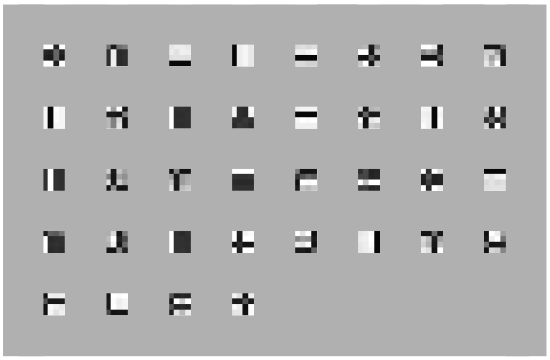


Fig. 3. The resulting basis vectors for standard NMF in the overcomplete case (number of allowed basis vectors 36). They are dominated by single-pixel solutions, because an optimal reconstruction can be achieved using them at the 16 different positions, if there are no further constraints on the activation.

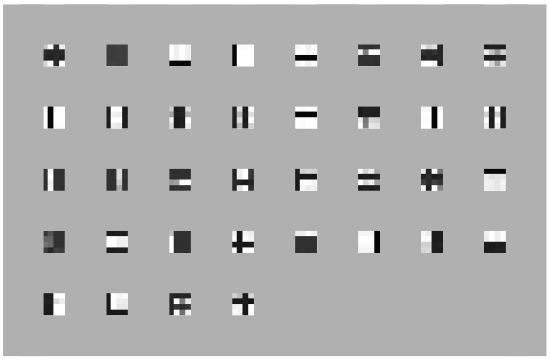
For an increased number of allowed basis vectors, the result of standard NMF is dominated by single-pixel solutions, because they allow a perfect reconstruction. The result for 36 basis vectors can be seen in fig. 3.

By increasing the sparsity term, the basis vector solutions become less fragmented, since the system tends to encode the input images using less basis vectors. One possible encoding would be to use all possible single and dual bar patterns, which amounts to a total of $8 + 8 * 7/2 = 36$ basis vectors, and to encode all input images in term of these patterns. In fig. 4, the results for the 36-basis-vector-encoding case are shown for sparse NMF using the multiplicative update rule (16). To verify the results, we also run simulations comparing the sparse NMF outcome with its corresponding gradient descent equivalent, and got indistinguishable results.

We see that standard NMF and sparse NMF produce qualitatively different solutions. The sparse NMF finds the single bar patterns but also dedicates additional basis vectors to the double-bar patterns, three-bar patterns, etc., since from the point of view of the sparse activation term in the cost function it is much cheaper to code them using a single basis vector than by combining several of them. We get a sequence of basis vectors of increasing "complexity": First the single-bar patterns appear, then the double-bar patterns, then more complex patterns.



λ_H=0.0



λ_H=0.25

Fig. 4. The resulting basis vectors for sparse NMF, using the multiplicative update rule (18). Shown are the results for 2 different sparsity parameter values.

The simulations were done with 250 input images and run for 1000 steps. The matrices W and H were initialized with equidistributed random values between 0.5 and 1.0.

B. Discussion

The inclusion of the sparsity term into the cost function (10) is a standard technique in the sparse coding framework, and the update rule for the activities (14) can be calculated directly from it. It is also straightforward to prove convergence of (14) with respect to (11) in a mathematically rigourous way if one follows closely the steps as indicated in [4]. Nevertheless, the problems with sparse NMF arise when the normalization of the basis vectors is considered. Here, we have shown how to merge sparsity constraints with NMF yet maintain the efficiency and ease of use of the NMF framework. The algorithm we presented does not depend on any learning rate parameter and exhibits fast convergence.

Because of the asymmetry introduced between W and H due to the basis vector normalization, the convergence proof [4] cannot be extended to the multiplicative update rule (16) in a straightforward way. In fact, it may not be guaranteed that (16) always decreases in terms of the cost function (11), nor that it finds all possible minima. Nevertheless, we have tested the algorithm extensively and the results always converged to sensible solutions. Our conjecture is that the reason for convergence is that the rescaling of the gradient introduced by the multiplicative update rule results in a modification step direction that has a positive projection on the true gradient (which is always true because of the non-negativity of W and H). Now, as long as the modification step size is sufficiently small (and this is true when the reconstructions R approach the input images V), we are on the safe side. To test this, we run simulations with a gradient descent update rule derived from (11) and obtained qualitatively and quantitatively similar results. In this case, we used standard parallel update for each basis vector $\mathbf{W}_j$, which resulted in very fast convergence.

Non-negative coding with sparsity constraints has already been explored to some extent for learning and modeling of receptive fields (see e.g. [1]). In particular, there are cases where an overcomplete representation of the data is unavoidable (for an example see e.g. [6]); then sparse coding becomes necessary to bypass trivial solutions for the basis vectors. We presented an algorithm that includes control over sparsity and that seamlessly extends the NMF method while retaining its advantages of parameter-independent gradient descent, fast convergence and easy numerical handling. We think that the presented combination of sparse coding and non-negative factorization may be potentially useful for a number of interesting applications.

Sparseness in the activities seems to be in contradiction to the parts-based representation (somehow suggestive of sparseness in the basis vectors) suggested by the NMF. Nevertheless, the parts-based character of basis vectors gained by NMF depends on (i) the statistics of the data (such as the sparse and localized basis vectors representing eyes, mouth and nose parts in the case of faces from [3]) (ii) the number of available basis vectors and (iii) the sparseness imposed on the activities (sparse activities implying non-sparse basis vectors and viceversa). For an overcomplete representation (in the sense that there is a larger number of basis vectors available than the dimensionality of the input space), NMF settles to the trivial solution with each basis vector representing a single pixel, as shown. In terms of activation patterns, this solution is not interesting since it basically mimics the input. To prevent this situation, a term was included into the cost function that favors sparser activation patterns and that allows to control the desired degree of sparsity in a seamless way. Nevertheless, the representation is still parts-based, in the sense that parts are additively put together to reconstruct the input.

REFERENCES

[1] P. O. Hoyer. Modeling receptive fi elds with non-negative sparse coding. In *Tenth Annual Computational Neuroscience Meeting (CNS)*, Chicago, Illinois, 2002. Conference proceedings paper.

[2] P. O. Hoyer. Non-negative sparse coding. In *NNSP 2002 conference proceedings*, 2002. Conference proceedings paper.

[3] D. D. Lee and H. S. Seung. Learning the parts of objects with nonnegative matrix factorization. *Nature*, 401:788–791, 1999.

[4] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 556–562. MIT Press, 2001.

[5] B. A. Olshausen. Emergence of simple-cell receptive fi eld properties by learning a sparse code for natural images. *Nature*, pages 607–609, 1996.

[6] H. Wersing, J. Eggert, and E. Körner. Sparse coding with invariance constraints. In *ICANN/ICONIP 2003 conference proceedings*, 2003. Conference proceedings paper.