Positive Tensor Factorization

Max Welling [1] and Markus Weber [2]

California Institute of Technology
[1] Dept. of Electrical Engineering
[2] Dept. of Computation and Neural Systems
Pasadena, CA 91125
USA
email:
[1] welling@vision.caltech.edu
[2] rmw@vision.caltech.edu

Abstract: *A novel fixed point algorithm for positive tensor factorization (PTF) is introduced. The update rules efficiently minimize the reconstruction error of a positive tensor over positive factors. Tensors of arbitrary order can be factorized, which extends earlier results in the literature. Experiments show that the factors of PTF are easier to interpret than those produced by methods based on the singular value decomposition, which might contain negative values. We also illustrate the tendency of PTF to generate sparsely distributed codes.*

Keywords: *PCA, SVD, postive matrix factorization, feature extraction*

1. Introduction

Principal component analysis (PCA) and singular value decomposition (SVD) are among the most widely used techniques for data analysis. Their function is to find a lower dimensional, orthonormal basis by minimizing the reconstruction error of the data.

It is often useful to interpret the eigenvectors produced by PCA as "modes" or "features" which are to be linearly combined to represent the data. The mode with the largest eigenvalue explains most of the variance (or energy) of the data. In this paper we argue that for intrinsically postive data, like the pixel values of color images, an interpretation of the eigenvectors is often impossible, since many eigenvectors must necessarily contain negative entries. This fact is easily seen by observing that the first eigenvector will be entirely postive for positive data, while subsequent eigenvectors must be oriented orthogonal to it. Thus, if the first eigenvalue does not contain many zeros, the remaining eigenvectors will contain negative entries. For purely illustrative purposes, we choose to reduce the number of color channels of the image in Figure 1a from 3 to 2, using both PTF (1b) and SVD (1c). While the factors produced by PTF can easily be interpreted as reflections and peel, the second SVD-factor is "meaningless" since the colorscheme is undefined for negative values.

Following [4] and [3], we propose to drop the orthogonality constraint in the linear factorization, and simply minimize the recontruction error directly under a positivity constraint. A second advantage of positive matrix factorization is explained in [3]. There, it was found that the positivity constraint has the tendency to generate sparse codings, i.e. many entries of the factors are zero. Sparse codings are of biological interest, since there is evidence that the brain codes information in this way [1]. Sparse coding is an intermediate between "grandmother-cell coding", where every datum is explained by one basis vector only (i.e. the data are clustered), and distributed coding where every datum is explained by a combination of a large set of basis vectors (e.g. PCA) [2]. Although, from a representational point of view, distributed coding needs far fewer basis vectors than grandmother-cell coding, efficient learning of distributed codes only seems feasible in linear models. However, linear transformations are often not sufficient to capture more complicated structure in the data. It would therefore be advantageous to have a simple, nonlinear method to produce meaningful distributed codes. Non-negative matrix factorization

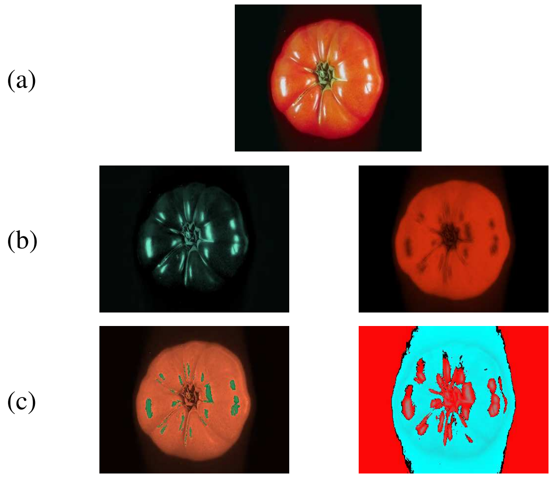

Figure 1. Decomposition of a color image in terms of two "color components". The image was represented as $X_{I,c} = A_{I,1}B_{c,1} + A_{I,2}B_{c,2}$, where I represents the image location indices (i,j) and c is the RGB color index. The two component images $A_{I,1}B_{c,1}$ and $A_{I,2}B_{c,2}$ are shown in b) for PTF and c) for SVD. The second SVD-factor is "meaningless" since the colorscheme is undefined for negative values.

[3] (NMF), positive matrix factorization [4] (PMF), and, as we will see, PTF partly achieve this goal. They tend to generate sparse codings (a few but not all basis vectors help explain the datum) while being slightly nonlinear due to the positivity constraint.

In this paper we present a novel algorithm that computes positive factors for tensors of arbitrary order by minimizing the reconstruction error.

2. PTF ALGORITHM

Consider a tensor of order d, $X_{i_1,\dots,i_d}$, with positive entries. We are interested in factorizing this tensor in terms of F positive components, $A^{(j)}_{i_j,a}$, in the following way,

$$X_{i_1,\dots,i_d} = \sum_{a=1}^{F} A^{(1)}_{i_1,a} A^{(2)}_{i_2,a} \dots A^{(d)}_{i_d,a}, \tag{1}$$

such that the reconstruction error,

$$RE = \sum_{i_1,\dots,i_d} \left(X_{i_1,\dots,i_d} - \sum_{a=1}^{F} A^{(1)}_{i_1,a} A^{(2)}_{i_2,a} \dots A^{(d)}_{i_d,a} \right)^2, \tag{2}$$

is minimized.

The following algorithm achieves this.

1) Initialize all $A^{(j)}_{i_j,a}$ by random positive values.
2) Iterate 3, 4, 5 until the reconstruction error (2) is changing less than a specified value.
3) Iterate 4, 5 for all indices $i_1,\dots,i_d$.
4) For index i_j,
   Define $Y_{i_j,I} = X_{i_1,\dots,i_d}$, grouping $i_1,\dots,i_{j-1},i_{j+1},\dots,i_d$ into one index I.
   Define $M_{I,a} = A^{(1)}_{i_1,a} \dots A^{(j-1)}_{i_{j-1},a} A^{(j+1)}_{i_{j+1},a} A^{(d)}_{i_d,a}$.
5) Update the component $A^{(j)}_{i_j,a}$ using the rule,

   $$A^{(j)} \to A^{(j)} \times \frac{Y \cdot M}{A^{(j)} \cdot M^T \cdot M}, \tag{3}$$

   while leaving the other components fixed. Here, '$\times$' denotes component-wise multiplication, '$\cdot$' denotes matrix multiplication and fractions are computed component-wise as well.
6) Calculate $K^{(j)}_a = \sqrt{\sum_{i_j} (A^{(j)}_{i_j,a})^2}$. Scale all factors to equal length using,

   $$A^{(j)}_{i_j,a} \to A^{(j)}_{i_j,a} \frac{\left(\prod_{i=1}^{d} K^{(i)}_a \right)^{\frac{1}{d}}}{K^{(j)}_a} \tag{4}$$

2.1. Essential Properties of the Fixed Point Algorithm

The update rule (3) can be understood by noticing that the following conditions are fullfilled,

I) At every iteration the $A^{(j)}_{i_j,a}$ are scaled by a positive scale factor $S^{(j)}_{i_j,a} \in [0,\infty]$ and therefore remain positive.

II) The minimum of the reconstruction error is a fixed point of the update rule.

III) The $A^{(j)}_{i_j,a}$ are changed in the direction of the negative gradient of the reconstruction error (measured locally in $A^{(j)}_{i_j,a}$). Note that this does not imply that the reconstruction error has actually decreased after an update.

Property I follows by observing that the scale factor in (3) is always positive if we initialize the $A^{(j)}_{i_j,a}$ with positive (random) numbers. Properties II and III are simply proved by inspection of the derivative of the reconstruction error as a function of one particular $A^{(j)}_{i_j,a}$,

$$\frac{\partial}{\partial A^{(j)}_{i_j,a}} \sum_{I,i_j} \left(Y - A^{(j)} \cdot M^T \right)^2_{i_j,I} = 2 \left(A^{(j)} \cdot M^T \cdot M - Y \cdot M \right)_{i_j,a}. \tag{5}$$

At the minimum of the reconstruction error the derivatives vanish, which implies that the scale factor is the identity. Therefore, the minimum is a fixed point of the update rule. Also, if the gradient is negative, the scale factor is greater than 1, which implies that we move towards the minimum. Reversely, if the gradient is positive, the scale factor is smaller than 1, which also implies that we move towards the minimum.

Notice that these properties do not imply the stability of the algorithm in the same sense as gradient descent is not guaranteed to improve a costfunction. However, not in any run of the algorithm have we encountered a single instance in which the reconstruction error actually increased or the algorithm got stuck in a cycle. But, in the absence of a convergence proof we formulate two ways which may be used when the reconstruction error fails to decrease during the execution of the algorithm.

2.2. Alternative Update Rules

One pitfall could be that the updated factor $A^{(j)}$ overshoots the minimum and smaller steps are required. To accomplish this we observe that all three properties are preserved if we use a general function, F, of the scale factor in (3),

$$S^{(j)}_{i_j,a} = F\left(\frac{Y \cdot M}{A^{(j)} \cdot M^T \cdot M} \right)_{i_j,a}, \tag{6}$$

where F has the following properties,

$$\begin{aligned} F(x) &> 0 \quad \forall \quad x > 0 \\ F(x) &= 1 \qquad\;\; x = 1 \\ F(x) &> 1 \quad \forall \quad x > 1 \\ F(x) &< 1 \quad \forall \quad x < 1 \end{aligned} \tag{7}$$

One such function is given by taking a positive power of the scale factor. Using a power smaller than 1 has the effect of taking smaller steps and decreasing the chance of overshooting the mimimum. It can be shown that there always exists a positive power, such that the fixed point is stable.

Another possibility to decrease the stepsize is to use gradient descent update rules [7]. The positivity constraint is enforced by reparametrizing $A^{(j)}_{i_j,a} = \exp(a^{(j)}_{i_j,a})$ and performing gradient descent on $a^{(j)}_{i_j,a}$. The update rules become,

$$\begin{aligned} a^{(j)} &\rightarrow a^{(j)} + \eta\, A^{(j)} \times (Y \cdot M - A^{(j)} \cdot M^T \cdot M), \\ A^{(j)} &= \exp(a^{(j)}), \end{aligned} \tag{8}$$

where $\times$ denotes again component-wise multiplication. To optimize convergence, one could "anneal" the stepsize η by decreasing its value as the minimum is approached.

2.3. Alternative Error Function

In [3] a different error function is minimized, based on the interpretation of non-negative matrix factorization (NMF) as a Poisson noise model. In our notation, and generalized to tensors of any order, this objective function would be,

$$\begin{aligned} RE &= \sum_{i_1,\ldots,i_d} X_{i_1,\ldots,i_d} \log\left(\sum_a A^{(1)}_{i_1,a} \ldots A^{(d)}_{i_d,a}\right) \\ &- \sum_{i_1,\ldots,i_d} \left(\sum_a A^{(1)}_{i_1,a} \ldots A^{(d)}_{i_d,a}\right). \end{aligned} \tag{9}$$

We can use our general procedure to derive update rules for this case as well. Define $Y_{j,I}$ and $M_{I,a}$ as above. Take derivatives with respect to $A^{(j)}_{i_j,a}$, which give a strictly postive and a strictly negative term. Define the scale factor as minus the negative term divided by the postive term. All three properties are then again guaranteed. This leads to a similar algorithm as described in this section, where the update rule is now given by,

$$A^{(j)}_{i_j,a} \rightarrow \frac{A^{(j)}_{i_j,a}}{\sum_J M_{J,a}} \sum_I \frac{Y_{i_j,I} M_{I,a}}{(A^{(j)} \cdot M^T)_{i_j,I}}. \tag{10}$$

For matrix factorization, this update rule coincides with the one given in [3] except for a scaling convention.

2.4. Generalizations

We would like to mention two other possible generalizations for which update rules can be easily derived in our formalism. In the experiment section we will see an example of a case where two indices of the matrix $X_{i_1,\ldots,i_d}$ are symmetric, i.e. X is invariant with respect to the interchange of those two indices. In that case we can set the two corresponding factors equal, $A^{(j)}_{i_j,a} = A^{(k)}_{i_k,a}$. Although it is easy to derive a general formula for this, we will only need the case of a symmetric *matrix* (two-tensor). It turns out that in this case we need to decrease the scalesize using a power of $\frac{1}{2}$. This leads to the following update rule,

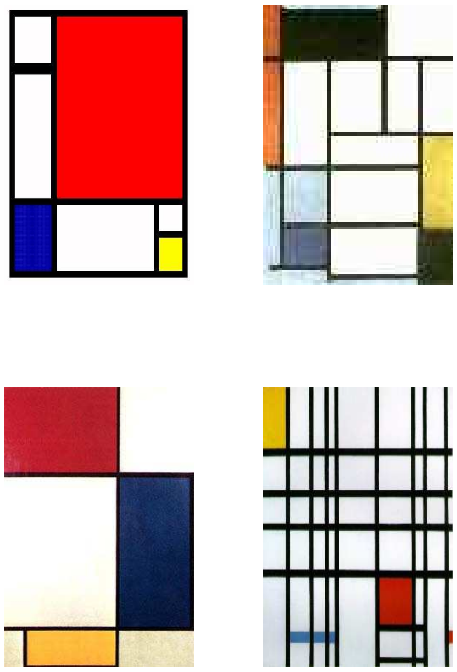

Figure 2. Four Mondriaan paintings used in the experiments.

$$A \rightarrow A \times \left(\frac{X \cdot A}{A \cdot A^T \cdot A}\right)^{\frac{1}{2}}. \tag{11}$$

Finally, we mention the possibility of partial factorizations, where tensors of higher order are factorized in tensors of arbitrary (but lower) order. We will not discuss the update rules here, but they can be derived straightforwardly using the general procedure described in this section.

3. Experiments

In the following we will describe three experiments to illustrate potential applications of PTF.

Experiment 1
In this experiment we used the 4 Mondriaan paintings shown in Figure 2. We cut out 4000 patches of size 15×15, and organized them in a data matrix, $X_{(i,j,c),n}$. The pixel-location indices, i, j, and the color index, c, were reshaped into one index I, while n labeled the patches. From this matrix, 25 basis images were calculated through the decomposition $X_{I,n} = \sum_{a=1}^{25} W_{I,a} \lambda_{n,a}$. The λ play the role of coefficients, which multiply the color basis images $W_{i,j,c}$ to reconstruct the original matrix X. Before plotting, we rescaled all basis images by the same factor, such that the maximum of the matrix W was 255 (this only effected their brightness). The results are shown in Figure 3. These results are contrasted with a palette calculated with SVD (Figure 4). It is clear that PTF generates sparser factors which have a more intuitive appeal than the global patterns of SVD

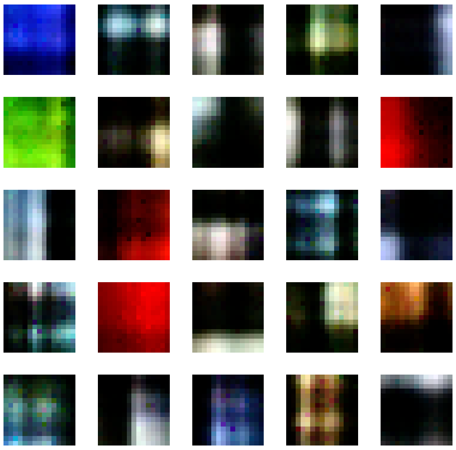

Figure 3. Palette of PTF factors for experiment 1.

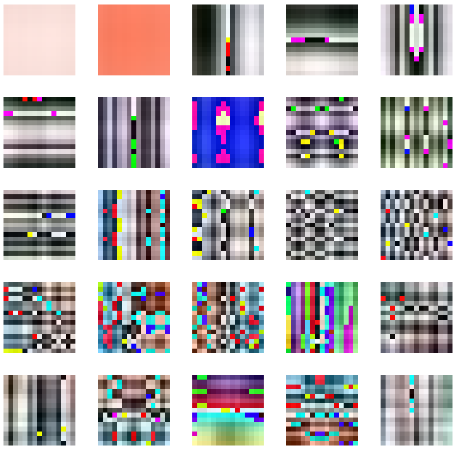

Figure 4. Palette of SVD factors for experiment 1

that has to assign arbitrary colors to negative numbers. This result verifies the findings in [3] obtained with black and white images.

Experiment 2

To test the factorization of a tensor in three factors, we generated 1000 patches of size 6×6 from the four Mondriaan paintings in Figure 2. This time we organized the data in a matrix $X_{i,(j,c),n}$, where only the horizontal pixel label j and the color index c were reshaped into a new index I. We factored this matrix as follows: $X_{i,I,n} = \sum_{a=1}^{16} V_{i,a} W_{I,a} \lambda_{n,a}$. For Mondriaan paintings, which only contain horizontal and vertical edges, this decomposition is not unreasonable. The result is contrasted with a 3-D pseudo-SVD algorithm [8], which basically contains the same steps as our PTF algorithm, but with an update rule that calculates the pseudo inverse at every iteration[1] $A \rightarrow Y \cdot M \cdot (M^T \cdot M)^{-1}$. We can conclude again that the PTF-palette shown in Figure 5 has more local

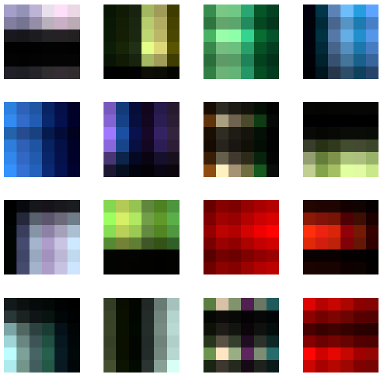

Figure 5. Palette of PTF factors for experiment 2.

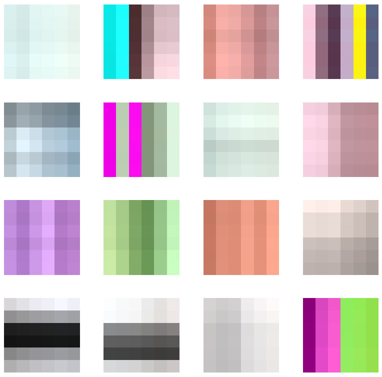

Figure 6. Palette of pseudo-SVD factors for experiment 2.

features than the pseudo-SVD palette shown in Figure 6. Notice that the basis images of a pseudo-SVD are not necessary orthogonal (as in SVD) which accounts for the more homogeneous appearance of the factors. The convergence of PTF is compared with the convergence of pseudo-SVD in Figure 7.

Experiment 3

In [5] a factorization approach to grouping is discussed. The technique is based on the factorization of an affinity matrix, which is, in our case, given by,

$$X_{IJ} = \exp\left(-\frac{\| \vec{\text{Image}}_I - \vec{\text{Image}}_J \|^2}{\sigma^2}\right), \qquad (12)$$

where $\vec{\text{Image}}_I$ represents an image in color space: I is the reshaped pixel location (i,j), and $\vec{\ }$ denotes a vector in RGB space. This matrix is then factorized as $X = A \cdot A^T$ (with X symmetric). In the original approach [5], only the first eigenvector of an SVD could be used to extract a foreground object, because only the entries of the first eigenvector are strictly positive for positive data. Our approach opens the possibility to exctract more groups simultaneously (using the updates (11)) as is shown in in Figure 8. Figure 8a shows the original subsampled Mondriaan painting, while 8b shows the 3 extracted groups, based on color, and defined by thresholding the factors

[1]Note the similarity between this update rule and the PTF update rule. In order to avoid the matrix inverse which generates negative entries, we use the component-wise division. For matrix factorization the pseudo-SVD algorithm is equivalent to the EM update rules found in [6]. Also, if we are looking for just one factor, $M^T \cdot M$ is scalar, which implies that PTF and pseudo-SVD coincide.

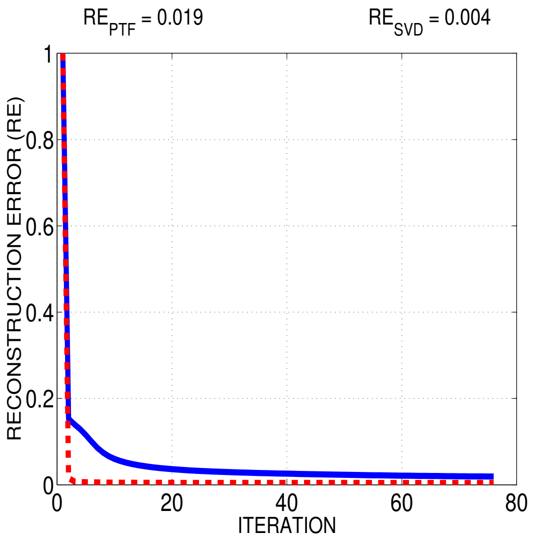

Figure 7. Convergence of PTF (blue line) and pseudo-SVD (dashed red line). The vertical scale is normalized such that the initial random guess has $RE = 1$.

(a)

(b)

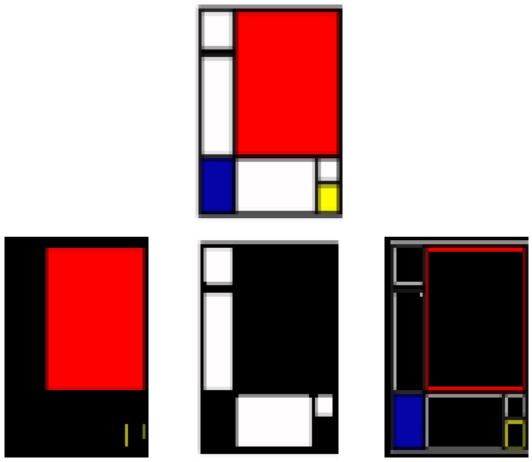

Figure 8. a) Original downsampled Mondriaan painting. b) Three extracted factors using PTF.

$A_{I,a}$. The fact that the third factor contains boundaries of different colors, besides the blue patch, is probably an artifact of the subsampling.

4. Discussion

In this paper we have described a new algorithm for factorizing a tensor of arbitrary order into positive factors. For intrinsically positive data, the advantage is that the factors are easier to interpret. It was also confirmed that the method produces sparser features than traditional SVD or PCA methods. In three experiments with Mondriaan paintings we tested the factorization algorithm for tensors of order 2 and 3, and compared it with a pseudo-SVD method. In another experiment we extended a factorization approach to grouping, described in [5], and showed that PTF allows the extraction of multiple groups.

5. Acknowledgements

We thank Pietro Perona, Maneesh Sahani, Luis Concalves for stimulating discussions. M. Welling would like to acknowledge the Sloan Center for its ongoing financial support.

References

[1] D.J. Field, "Relation between the Statistics of Natural Images and the Response Properties of Cortical Cells", *Journal Optical Society America A*, 1987, pp. 2370-2393.

[2] G.E. Hinton and Z. Ghahramani, "Generative Models for Discovering Sparse Distributed Representations", *Philosophical Transactions of the Royal Society B*, 1997, pp. 1177-1190.

[3] D.D. Lee and H.S. Seung, "Learning the Parts of Objects by Non-Negative Matrix Factorization", *Nature*, 1999, pp. 788-791.

[4] P. Paatero and U. Tapper, "Positive Matrix Factorization - a Nonnegative Factor Model with Optimal Utilization of Error-Estimates of Data Values" *Environmetrics*, 1994, pp. 111-126.

[5] P. Perona and W. Freeman, "A Factorization Approach to Grouping", *California Institute of Technology, Electrical Engineering*, 1998.

[6] S. Roweis, "EM Algorithms for PCA and Sensible PCA", *California Institute of Technology, Computation and Neural Systems*, 1997.

[7] M. Sahani, personal communication, 1999

[8] D. Shy and P. Perona, "X-Y Separable Pyramid Steerable Scalable Kernels", in *Proceedings CVPR*, Seattle, 1994.