ELSEVIER

Pattern Recognition Letters 22 (2001) 1255–1261

Pattern Recognition Letters

www.elsevier.com/locate/patrec

Positive tensor factorization

Max Welling *, Markus Weber

California Institute of Technology 136-93, Pasadena, CA 91125, USA

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. © 2001 Published by Elsevier Science B.V.

Keywords: PCA; SVD; Positive 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 positive 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 positive 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 Fig. 1(a) from 3 to 2, using both positive tensor factorization (PTF) (Fig. 1(b)) and SVD (Fig. 1(c)). 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 Paatero and Tapper (1997) and Lee and Seung (1999), we propose to drop the orthogonality constraint in the linear factorization, and simply minimize the reconstruction error directly under a positivity constraint. A second advantage of positive matrix factorization is explained in (Lee and Seung, 1999). 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

* Corresponding author.

E-mail addresses: welling@vision.caltech.edu (M. Welling), rmw@vision.caltech.edu (M. Weber).

0167-8655/01/$ - see front matter © 2001 Published by Elsevier Science B.V.
PII: S0167-8655(01)00070-8

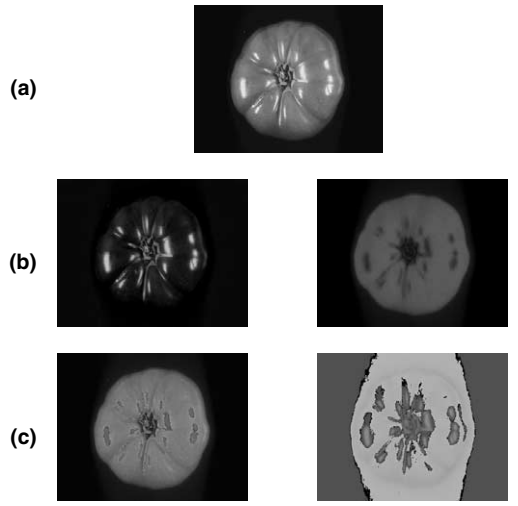

Fig. 1. Decomposition of a color image in terms of two "color components". The image (a) was represented as $X_{I,c} = A_{I,1}B_{c,1} + A_{I,2}B_{c,2}$, where I represents the image location indices (i, j) and c is the RGB color index. The two component images $A_{I,1}B_{c,1}$ and $A_{I,2}B_{c,2}$ are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

brain codes information in this way (Field, 1987). 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 combining contributions from all basis vectors (e.g. PCA) (Hinton and Ghahramani, 1997). 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 (NMF, Lee and Seung, 1999), positive matrix factorization (PMF, Paatero and Tapper, 1997) 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, $X_{i_1,\ldots,i_d}$, 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:

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

such that the reconstruction error,

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

is minimized.

The following algorithm achieves this:

1. Initialize all $A^{(j)}_{i_j,a}$ by random positive values.
2. Iterate 3–5 until the reconstruction error (2) is changing less than a specified value.
3. Iterate 4 and 5 for all indices $i_1, \ldots, i_d$.
4. For index i_j,
   Define $Y_{i_j,I} = X_{i_1,\ldots,i_d}$, grouping $i_1, \ldots, i_{j-1}, i_{j+1}, \ldots, i_d$ into one index I.
   Define $M_{I,a} = A^{(1)}_{i_1,a} \cdots A^{(j-1)}_{i_{j-1},a} A^{(j+1)}_{i_{j+1},a} A^{(d)}_{i_d,a}$.
5. Update the component $A^{(j)}_{i_j,a}$ using the rule,
$$A^{(j)} \to A^{(j)} \times \frac{Y \cdot M}{A^{(j)} \cdot M^{\mathrm{T}} \cdot M} \tag{3}$$
   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,
$$A^{(j)}_{i_j,a} \to A^{(j)}_{i_j,a} \frac{\left(\prod_{i=1}^{d} K^{(i)}_a \right)^{1/d}}{K^{(j)}_a}. \tag{4}$$

2.1. Essential properties of the fixed point algorithm

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

(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 $A^{(j)}_{i_j,a}$,

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

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),

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

where F has the following properties,

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

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 minimum. 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 (Sahani, personal communication). 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,

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

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 (Lee and Seung, 1999) 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,

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

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 positive and a strictly negative term. Define the scale factor as minus the negative term divided by the positive 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

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

For matrix factorization, this update rule coincides with the one given in (Lee and Seung, 1999) 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 scale-size using a power of $\frac{1}{2}$. This leads to the following update rule:

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

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.

3.1. Experiment 1

In this experiment we used the four Mondriaan paintings shown in Fig. 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 Fig. 3. These results are contrasted with a palette calculated with SVD (Fig. 4). It is clear that PTF generates sparser factors which have a more intuitive appeal than the global patterns of SVD that has to assign arbitrary colors to negative numbers. This result verifies the findings in (Lee and Seung, 1999) obtained with black and white images.

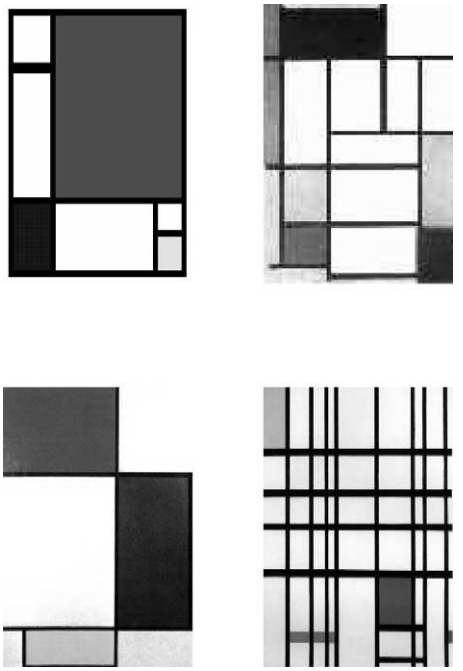

Fig. 2. Four Mondriaan paintings used in the experiments.

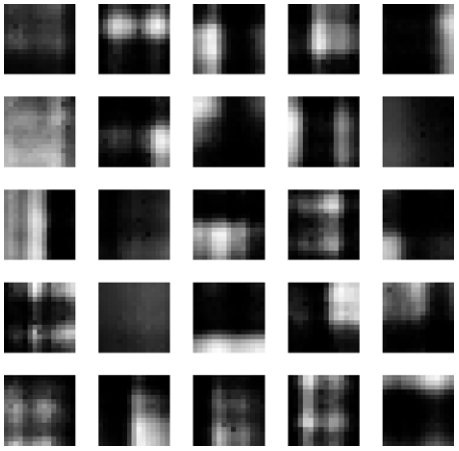

Fig. 3. Palette of PTF factors for experiment 1.

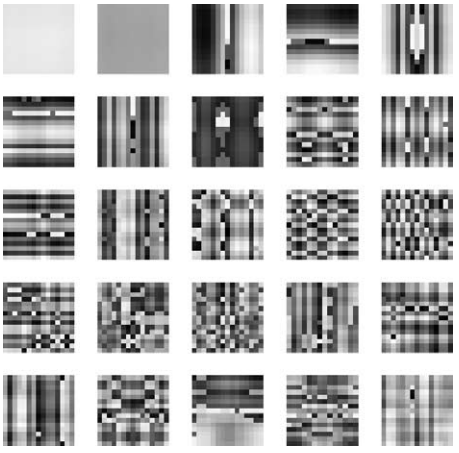

Fig. 4. Palette of SVD factors for experiment 1.

3.2. 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 Fig. 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

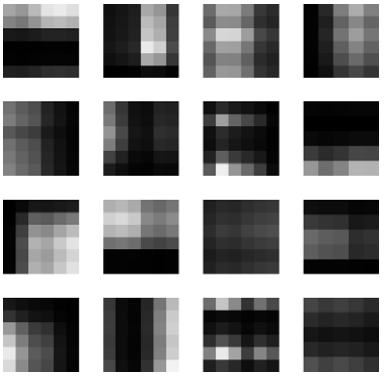

Fig. 5. Palette of PTF factors for experiment 2.

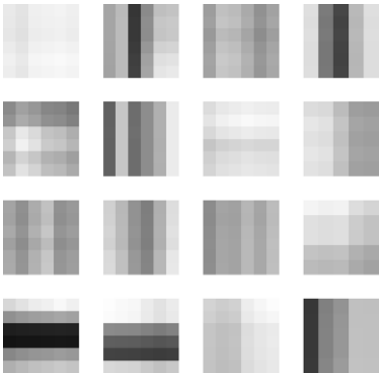

Fig. 6. Palette of pseudo-SVD factors for experiment 2.

(Shy and Perona, 1994), which basically contains the same steps as our PTF algorithm, but with an update rule that calculates the pseudo inverse at every iteration [1] $A \rightarrow Y \cdot M \cdot (M^{\mathrm{T}} \cdot M)^{-1}$. We can conclude again that the PTF-palette shown in Fig. 5 has more local features than the pseudo-SVD palette shown in Fig. 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

[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 (Roweis, 1997). Also, if we are looking for just one factor, $M^{\mathrm{T}} \cdot M$ is scalar, which implies that PTF and pseudo-SVD coincide.

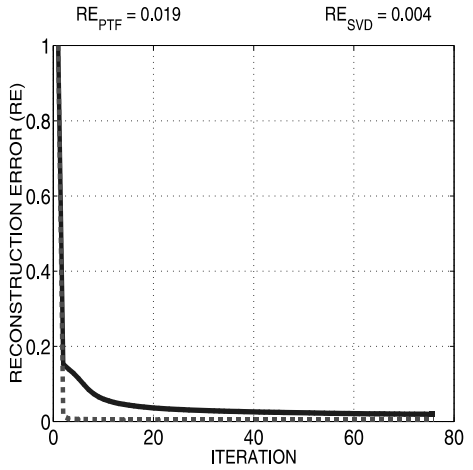

Fig. 7. Convergence of PTF (—) and pseudo-SVD (- - - -). The vertical scale is normalized such that the initial random guess has RE = 1.

convergence of PTF is compared with the convergence of pseudo-SVD in Fig. 7.

3.3. Experiment 3

In (Perona and Freeman, 1998) 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,

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

where $\overrightarrow{\text{Image}}_I$ 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 $X = A \cdot A^{\mathrm{T}}$ (with X symmetric). In the original approach (Perona and Freeman, 1998), 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 Fig. 8. Fig. 8(a) shows the original subsampled Mondriaan painting, while Fig. 8(b) shows the three extracted groups, based on color, and defined by thresholding the factors $A_{I,a}$. 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

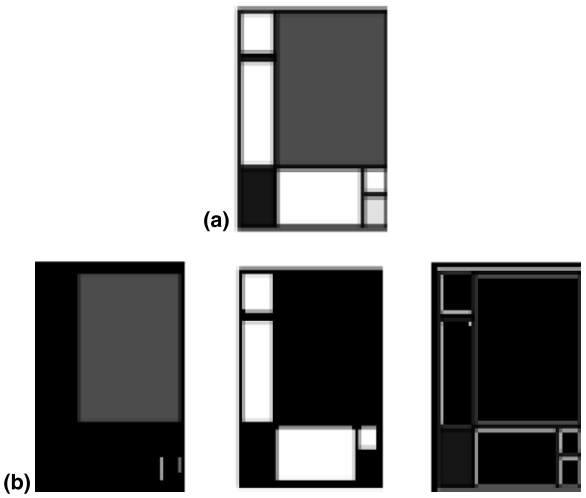

Fig. 8. (a) Original downsampled Mondriaan painting. (b) Three extracted factors using PTF.

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. It would be interesting to compare the levels of sparsity achieved by PTF with techniques like ICA, which are also known to produce sparse representations.

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 (Perona and Freeman, 1998), and showed that PTF allows the extraction of multiple groups.

Acknowledgements

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

References

Field, D.J., 1987. Relation between the statistics of natural images and the response properties of cortical cells. J. Opt. Soc. Am. A 4, 2370–2393.

Hinton, G.E., Ghahramani, Z., 1997. Generative models for discovering sparse distributed representations. Philos. Trans. Roy. Soc. B 353, 1177–1190.

Lee, D.D., Seung, H.S., 1999. Learning the parts of objects by non-negative matrix factorization. Nature 401, 788–791.

Paatero, P., Tapper, U., 1997. Least squares formulation of robust non-negative factor analysis. Chemometr. Intell. Lab. 37, 23–35.

Perona, P., Freeman, W.T., 1998. A factorization approach to grouping. In: Proc. Euro. Conf. Comput. Vision, Freiburg, Germany, pp. 655–670.

Roweis, S.T., 1997. EM algorithms for PCA and sensible PCA. Neural Inf. Proc. Syst. 10, 626–632.

Shy, D., Perona, P., 1994. X–Y separable pyramid steerable scalable kernels. In: Proc. Conf. Comput. Vision Pattern Recognition, Seattle, US, pp. 237–244.