Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 40 (2007) 993–1003
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/pr

Handwritten digit classification using higher order singular value decomposition

Berkant Savas*, Lars Eldén

Department of Mathematics, Linköping University, 581 83 Linköping, Sweden

Received 5 October 2005; accepted 9 August 2006

Abstract

In this paper we present two algorithms for handwritten digit classification based on the higher order singular value decomposition (HOSVD). The first algorithm uses HOSVD for construction of the class models and achieves classification results with error rate lower than 6%. The second algorithm uses the HOSVD for tensor approximation simultaneously in two modes. Classification results for the second algorithm are almost down at 5% even though the approximation reduces the original training data with more than 98% before the construction of the class models. The actual classification in the test phase for both algorithms is conducted by solving a series least squares problems. Considering computational amount for the test presented the second algorithm is twice as efficient as the first one.

Keywords: Handwritten digit classification; Tensors; Higher order singular value decomposition; Tensor approximation; Least squares

1. Introduction

The automatic classification of handwritten digits is often considered as a standard problem in pattern recognition, and it involves many of the difficulties encountered in this area. The task of assigning an unknown object to one of the 10 predefined classes is a hard problem to solve, since the variation of the objects within each class is high, at the same time as objects from different classes may be quite similar.

There are many different approaches to solving this problem: principal component analysis (PCA), support vector methods, nearest neighbor and *k*-nearest neighbor methods, regression, statistical modelling and neural networks. Surveys of different pattern recognition methods are given in Refs.[1,2]. A comparison of different algorithms for classification of handwritten digits is given in Ref. [3]. The best performing algorithms are based on neural networks [4], and the so-called tangent distance [5], that uses a distance measure invariant under local affine transformations. Other algorithms are given in Ref. [6] (extended tangent distance), [7] (elastic matching) and [8] (manifold modelling). In general algorithms with good performance have either large descriptive complexity or are computationally heavy (in training and/or classification).

In this paper we will present two simple and efficient algorithms with fairly good performance. Both algorithms are based on the higher order singular value decomposition (HOSVD) of a tensor [9]. The first algorithm uses HOSVD to compute a small set of basis matrices that span the dominant subspace for each class of digits. The basis matrices are then used to describe unknown digits. This algorithm is closely related to SIMCA [10] and PCA. The second algorithm uses HOSVD to compress the training set. The class models (here basis vectors) are computed using the reduced data only, and classification is performed as in the first algorithm. The advantages are twofold; the descriptions of the class models require less memory and the classification phase is more efficient without drawbacks in the performance. The algorithm gives an error rate of 5% even after a compression of the training set with more than 98%.

In recent years the application of tensor methods to problems in pattern recognition and other areas has attracted

* Corresponding author. Tel.: +46 13 281496; fax: +46 13 135063.
E-mail addresses: besav@math.liu.se (B. Savas), laeld@math.liu.se (L. Eldén).

0031-3203/$30.00
doi:10.1016/j.patcog.2006.08.004

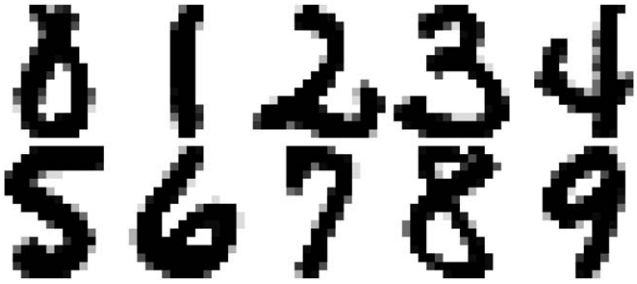

Fig. 1. Samples of handwritten digits from the US Postal Service database.

more and more attention. By tensors we mean multidimensional or multimode arrays. Often the data have a multidimensional structure and it is then somewhat unnatural to organize them as matrices or vectors. As a simple example, consider a time series of images. Each image is a two-dimensional data array and together with images from different time steps the data constitute a third order tensor (three-dimensional data array). In many cases it is beneficial to use the collected data without destroying its inherent multidimensional structure. Tensor methods have been used for a long time in chemometrics and psychometrics (see, e.g., [11]). Recently HOSVD has been applied to face recognition [12].

In this paper we use handwritten digits from the US Postal Service database to test the performance of the algorithms proposed. The digits are represented by 16×16 gray scale pixel images scanned from envelopes. This data set is widely used for evaluation of classification algorithms. Samples are shown in Figure 1.

The rest of the paper is structured as follows: Section 2 contains an introduction to tensor concepts and some theoretical results that are used in the algorithms. In Section 3 both algorithms are presented. The numerical tests are described in Section 4, where also a more detailed description of the data set is given.

The algorithms will be illustrated using pseudo-MATLAB code. Thus, in code examples we will use the notation `A(i,j,k)` meaning a_{ijk}. Also in formulas we will sometimes use MATLAB-style notation. For instance, we define the *mode* $-$ 1 *fibers* of a 3-tensor $\mathscr{A}$ to be the column vectors $\mathscr{A}(:, j, k)$. The definition of fibers along the other modes is obvious. Thus, fibers are characterized by fixing the index in all modes but one. Similarly, we define *slices* of a tensor to be the sub-tensors obtained by fixing the index in one mode, e.g. $\mathscr{A}(:, :, k)$.

2. Tensor concepts

Loosely speaking, an Nth order tensor is an object with N indices. The "dimensions" of a tensor will be referred to as modes. Vectors and matrices can be considered as first and second order tensors, respectively. In the applications of this paper we will deal with the case $N = 3$. Therefore, for simplicity, some of the theory presented in this section is stated only for third order tensors

$$\mathscr{A} \in \mathbb{R}^{I\times J\times K},$$

where I, J, K are positive integers; the vector space $\mathbb{R}^{I\times J\times K}$ has dimension IJK. The generalization to tensors of arbitrary order is obvious.

Let $\mathbb{R}^{I\times J\times K}$ be endowed with the usual Euclidean geometry. The scalar product $\langle \mathscr{A}, \mathscr{B}\rangle$ of two tensors $\mathscr{A}, \mathscr{B} \in \mathbb{R}^{I\times J\times K}$ is defined

$$\langle \mathscr{A}, \mathscr{B}\rangle = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} a_{ijk}b_{ijk}. \tag{1}$$

Two tensors $\mathscr{A}$ and $\mathscr{B}$ are said to be orthogonal if their scalar product is equal to zero

$$\langle \mathscr{A}, \mathscr{B}\rangle = 0. \tag{2}$$

The norm of a tensor $\mathscr{A}$ is defined

$$\|\mathscr{A}\| = \sqrt{\langle \mathscr{A}, \mathscr{A}\rangle}. \tag{3}$$

The scalar product and norm of vectors and matrices are defined analogously.

It is sometimes convenient to rearrange the elements of a tensor so that they form a matrix. We will refer to this as *matricizing* the tensor.[1] The mode$-n$ matricizing of a tensor $\mathscr{K}$ is an operation where the mode$-n$ fibers of $\mathscr{K}$ are aligned as the columns of a matrix, denoted $K_{(n)}$. One may assume that the columns of $K_{(n)}$ are ordered in a *forward cyclic* fashion, c.f. Ref. [14]. Then the matricizing of a 3$-$tensor $\mathscr{A} \in \mathbb{R}^{I\times J\times K}$ is defined by specifying where an element in $\mathscr{A}$ is placed in the matrix,

$$\begin{aligned}
&\mathbb{R}^{I\times JK} \ni A_{(1)} : a_{ijk} = a^{(1)}_{iv}, \quad && v = j + (k-1)K,\\
&\mathbb{R}^{J\times IK} \ni A_{(2)} : a_{ijk} = a^{(2)}_{jv}, \quad && v = k + (i-1)I,\\
&\mathbb{R}^{K\times IJ} \ni A_{(3)} : a_{ijk} = a^{(3)}_{kv}, \quad && v = i + (j-1)J.
\end{aligned}$$

Note that the column vectors of $A_{(n)}$ are the mode$-n$ fibers of $\mathscr{A}$.

There is no unique way of generalizing the rank concept for higher order tensors from the definitions of rank for matrices. One possibility is to define the n-rank of a tensor $\mathscr{A}$ as the dimension of the subspace spanned by the n-mode vectors, i.e.

$$\text{rank}_n(\mathscr{A}) = \text{rank}(A_{(n)}), \tag{4}$$

where $A_{(n)}$ is the matricized $\mathscr{A}$ along the *nth* mode, and "rank" without subscript denotes matrix rank. It is easy to

[1] Alternative terms are *unfolding* [9] or *flattening* [13].

verify that different n-ranks of a third order (and higher) tensor are usually not equal as is the case for matrices.

We now give the general definition of tensor–matrix multiplication.[2]

Definition 1 (*n–mode tensor–matrix multiplication*). Let $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $F \in \mathbb{R}^{J_n \times I_n}$. Then the $n-$mode tensor–matrix product of $\mathcal{A}$ and F is denoted $\mathcal{A} \times_n F$, and is defined by

$$(\mathcal{A} \times_n F)(i_1, \ldots, i_{n-1}, j_n, i_{n+1}, \ldots, i_N) = \sum_{i_n=1}^{I_n} \mathcal{A}(i_1, \ldots, i_N) F(j_n, i_n).$$

For instance, 1-mode multiplication of the tensor $\mathcal{A} \in \mathbb{R}^{I \times J \times K}$ by the matrix $F \in \mathbb{R}^{L \times I}$ is given by

$$\mathbb{R}^{L \times J \times K} \ni \mathcal{B} = \mathcal{A} \times_1 F, \quad \mathcal{B}(l, j, k) = \sum_{i=1}^{I} \mathcal{A}(i, j, k) F(l, i).$$

Given the tensor $\mathcal{A} \in \mathbb{R}^{I \times J \times K}$ and the matrices $F \in \mathbb{R}^{L \times I}$, $G \in \mathbb{R}^{M \times J}$ and $H \in \mathbb{R}^{N \times L}$ the tensor–matrix multiplication satisfies the following properties,[3]

$$(\mathcal{A} \times_1 F) \times_2 G = (\mathcal{A} \times_2 G) \times_1 F = \mathcal{A} \times_1 F \times_2 G \in \mathbb{R}^{L \times M \times K}, \tag{5}$$

$$(\mathcal{A} \times_1 F) \times_1 H = \mathcal{A} \times_1 (HF) \quad \in \mathbb{R}^{N \times J \times K}. \tag{6}$$

Since the column vectors of $A_{(n)}$ are the n-mode fibers of $\mathcal{A}$, it follows that n-mode multiplication $\mathcal{B} = \mathcal{A} \times_n X$ can be expressed in terms of the matricized tensor: the ordinary matrix multiplication

$$B_{(n)} = X A_{(n)}, \tag{7}$$

is followed by a reorganization of $B_{(n)}$ to the tensor $\mathcal{B}$.

2.1. Higher order SVD

The singular value decomposition (SVD) of a matrix is a very useful tool in many applications. Without loss of generality we assume that $F \in \mathbb{R}^{m \times n}$, with $m \geqslant n$.

Theorem 2 (*Matrix SVD*). *Every matrix $F \in \mathbb{R}^{m \times n}$ can be written as the product*

$$F = U \Sigma V^T, \tag{8}$$

where $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are orthogonal matrices and Σ is an $(m \times n)$ diagonal matrix with non-negative entries, ordered in the following way:

$$\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_n \geqslant 0. \tag{9}$$

The columns of U and V are called the left and right singular vectors, respectively, and the σ_i are the singular values.

A proof of the SVD theorem can be found e.g. in Ref. [16, Section 2.5.3]. Considering the matrix as a second order tensor we can express the SVD in terms of the n-mode product

$$F = \Sigma \times_1 U \times_2 V. \tag{10}$$

One possible generalization of the SVD to tensors was given in Ref. [9], and it is called the HOSVD.[4] We state the result for a 3-tensor $\mathcal{A}$.

Theorem 3 (*HOSVD [9]*). *A third order tensor $\mathcal{A} \in \mathbb{R}^{I \times J \times K}$ can be written as the product*

$$\mathcal{A} = \mathcal{S} \times_1 U \times_2 V \times_3 W, \tag{11}$$

with the following properties:

(1) *$U \in \mathbb{R}^{I \times I}$, $V \in \mathbb{R}^{J \times J}$ and $W \in \mathbb{R}^{K \times K}$ are orthogonal matrices,*
(2) *$\mathcal{S}$ is a real tensor of the same dimensions as $\mathcal{A}$ and satisfies*
   (a) (*all-orthogonality*) *any two different slices fixed in the same mode are orthogonal,*

$$\begin{aligned} &\langle \mathcal{S}(v, :, :), \mathcal{S}(\lambda, :, :) \rangle = 0 \quad \text{when } v \neq \lambda, \\ &\langle \mathcal{S}(:, v, :), \mathcal{S}(:, \lambda, :) \rangle = 0 \quad \text{when } v \neq \lambda, \\ &\langle \mathcal{S}(:, :, v), \mathcal{S}(:, :, \lambda) \rangle = 0 \quad \text{when } v \neq \lambda. \end{aligned} \tag{12}$$

   (b) (*ordering*) *the norms of the slices along every mode are ordered, e.g., for the first mode we have*

$$\|\mathcal{S}(1, :, :)\| \geqslant \|\mathcal{S}(2, :, :)\| \geqslant \cdots \geqslant 0. \tag{13}$$

The norms in (13) are in fact the singular values, $\sigma_i^{(n)}$, of the matricized tensors $A_{(n)}$.

The ordering property (13) implies that, loosely speaking, the "energy" or "mass" of the core tensor $\mathcal{S}$ is concentrated in the vicinity of the $(1, 1, 1)$ corner. This is the property that makes it possible to use the HOSVD for data compression.

The computation of the HOSVD of a tensor $\mathcal{A} \in \mathbb{R}^{I \times J \times K}$ can be done by separately computing the orthogonal

[2] Actually, we define *contravariant* tensor–matrix multiplication, see, e.g., [15, Section 1.2]. Since we will not deal with *covariant* tensor–matrix multiplication in this paper, we do not emphasize the distinction between the two variants.

[3] Both properties are valid in all modes of the tensor.

[4] The HOSVD is closely related to the Tucker model first introduced in psychometrics [17].

matrices U, V, and W, as the left singular matrices of $A_{(n)}$, $n = 1, 2, 3$.

(1) Compute the three SVDs:

$$A_{(1)} = US^{(1)}(V^{(1)})^T,$$
$$A_{(2)} = VS^{(2)}(V^{(2)})^T,$$
$$A_{(3)} = WS^{(3)}(V^{(3)})^T$$

without forming the $V^{(j)}$'s explicitly.

(2) Compute the core tensor: $\mathcal{S} = \mathcal{A} \times_1 U^T \times_2 V^T \times_3 W^T$.

Note that by avoiding the computation of the right singular matrices, one can take advantage of the common situation when, for instance, $I \ll JK$, and save a large number of floating point operations.

2.2. Tensor approximation by means of HOSVD

In the case of matrices data compression is very often done by low rank approximation: the best rank k approximation, in any orthogonally invariant norm, of a given matrix F is obtained by the SVD [18] [16, Theorem 2.5.3], i.e.

$$F = U\Sigma V^T \approx U_k \Sigma_k V_k^T,$$

where $\Sigma_k = \mathrm{diag}(\sigma_1, \ldots, \sigma_k)$ and U_k and V_k are the first k columns of U and V, respectively. The approximation is illustrated symbolically in Figure 2.

Tensor approximation can be done by HOSVD in an analogous way, due to the ordering property (13). However, for general tensors the approximation is not optimal in the norm (3) [19]. For third order tensors the approximation can be written as

$$\mathbb{R}^{I\times J\times K} \ni \mathcal{A} = \mathcal{S} \times_1 U \times_2 V \times_3 W \approx \tilde{\mathcal{S}} \times_1 \tilde{U} \times_2 \tilde{V} \times_3 \tilde{W}, \quad (14)$$

where $\tilde{\mathcal{S}} \in \mathbb{R}^{k_1\times k_2\times k_3}$, $\tilde{U} \in \mathbb{R}^{I\times k_1}$, $\tilde{V} \in \mathbb{R}^{J\times k_2}$, and $\tilde{W} \in \mathbb{R}^{K\times k_3}$. A measure of the quality of the approximation can be achieved by examining the decay of the n-mode singular values of $\mathcal{A}$ [9, Property 10]: the approximation error is small if the norm of the omitted part of the core $\mathcal{S}$ is small. We illustrate the low rank approximation of a tensor in Figure 3.

2.3. Orthogonal basis matrices

A matrix F can be written as a sum of rank one matrices in terms of the SVD (8)

$$F = U\Sigma V^T = \sum_{i=1}^{n} \sigma_i u_i v_i^T. \quad (15)$$

A similar decomposition can be achieved for third order tensors,

$$\mathcal{A} = \sum_{\nu=1}^{K} A_\nu \times_3 w_\nu, \quad (16)$$

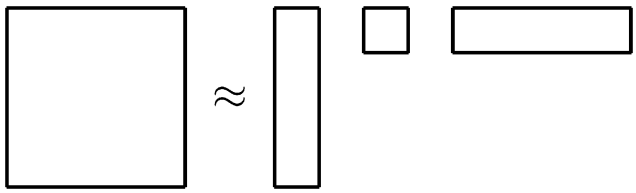

Fig. 2. Illustration of the SVD approximation.

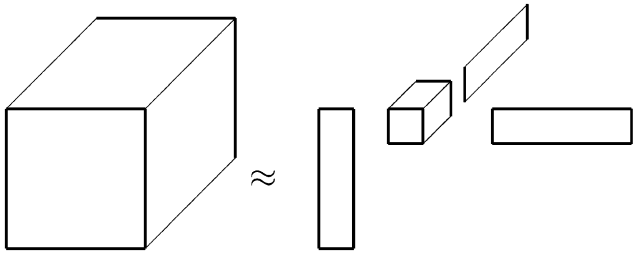

Fig. 3. Illustration of the tensor approximation.

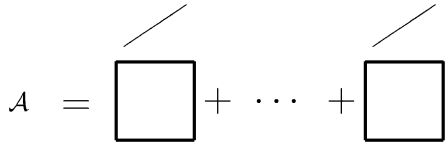

Fig. 4. Illustration of the decomposition in Eq. (16).

where $A_\nu = \mathcal{S}(:,:,\nu) \times_1 U \times_2 V$. Note that $\times_3$ is an *outer product* between the matrix A_ν and the vector w_ν yielding a 3-tensor.

Due to the all-orthogonality of $\mathcal{S}$, the A_ν are orthogonal:

$$\begin{aligned}\langle A_\nu, A_\mu\rangle &= tr(A_\nu^T A_\mu) = tr((U\mathcal{S}(:,:,\nu)V^T)^T \\ &\quad (U\mathcal{S}(:,:,\mu)V^T)) \\ &= tr(V\mathcal{S}(:,:,\nu)^T \mathcal{S}(:,:,\mu)V^T) \\ &= \langle \mathcal{S}(:,:,\nu), \mathcal{S}(:,:,\mu)\rangle = 0,\end{aligned}$$

when $\nu \neq \mu$ ("tr" in the calculation above denotes *trace*). These orthogonal matrices can be interpreted as a set of linearly independent basis matrices.

An illustration is given in Figure 4. The squares represent the A_ν and the tilted lines above them are the w_ν vectors representing third mode outer product.

3. Algorithms

3.1. Algorithm 1: Classification by HOSVD

In this section we describe how HOSVD can be used to build algorithms for handwritten digit classification. The training set of the digits is manually classified.

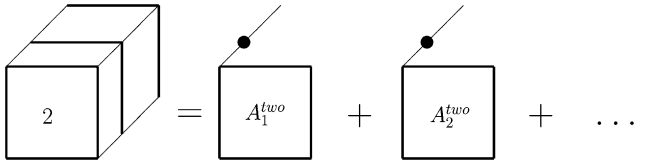

Fig. 5. Illustration of the decomposition in Eq. (17).

Considering each digit[5] as a point in $\mathbb{R}^{20\times 20}$, it is reasonable to assume that the digits of the training set constitute 10 fairly well separated clusters, because otherwise any classification algorithm would perform badly. At the same time, the "dominating" vectors in each cluster span a subspace in $\mathbb{R}^{400}$. We use a variant of the SIMCA algorithm [10] where one constructs a small set of orthogonal basis matrices (vectors) for each class spanning the dominant parts of the corresponding subspaces. We then determine which of the bases describes an unknown digit in the best way, i.e. we compute the approximation error for each of the 10 bases. The computation of the different sets of basis matrices is implemented using the HOSVD.[6]

3.1.1. Training phase

The set of orthogonal basis matrices is constructed as described in Section 2.3. The basis for each class is computed from a third order tensor with all the training digits of the same kind. Let $\mathcal{A}^{two} \in \mathbb{R}^{20\times 20\times K}$ be the tensor with the two's and assume that the HOSVD is computed. Then from Eq. (16)

$$\mathcal{A}^{two} = \sum_{v=1}^{K} A_v^{two} \times_3 w_v^{two}, \tag{17}$$

where the A_v^{two} are orthogonal basis matrices. This also means that every digit in $\mathcal{A}^{two}$ is a unique linear combination of the same basis matrices A_v^{two}. The coefficients in the linear combination are given by the elements of the w_v^{two} vectors. We can illustrate (17) as in Figure 5 where an arbitrary two in $\mathcal{A}^{two}$ is a linear combination of the A_v^{two}. The coefficients are symbolized as dots on the third mode vectors.

We truncate the sum in expression (17) to obtain a small and dominant k-dimensional subspace for each digit cluster. Suppose that the bases are constructed for all the classes, each comprising k basis matrices, and denote

$$T^{\mu} = (A_v^{\mu})_{v=1}^{k}, \quad \mu = 0, 1, \ldots, 9.$$

In addition, assume that the basis matrices are normalized. Thus,

$$\langle A_v^{\mu}, A_{\lambda}^{\mu} \rangle = \delta_{v\lambda}, \quad \mu = 0, 1, \ldots, 9,$$

where $\delta_{v\lambda}$ is the Kronecker delta. Let D be an unknown digit that is assumed to be normalized, $\|D\| = 1$. Which set of basis matrices T^{μ} is then describing D in the best way?

3.1.2. Test phase

Consider the minimization problem

$$\min_{\alpha_v^{\mu}} \|D - \sum_{v=1}^{k} \alpha_v^{\mu} A_v^{\mu}\|, \quad \mu \quad \text{fixed class index} \tag{18}$$

in which the α_v^{μ} are unknown scalars to be determined. This is a least squares problem, and it is easy to solve since the A_v^{μ} are orthonormal for μ fixed: the solution is given by

$$\hat{\alpha}_v^{\mu} = \langle D, A_v^{\mu} \rangle, \quad v = 1, 2, \ldots, k.$$

It is of some interest to note that α_1^{μ} is the cosine of the angle between the two matrices D and A_1^{μ},

$$\hat{\alpha}_1^{\mu} = \cos(\theta^{\mu}) = \langle D, A_1^{\mu} \rangle. \tag{19}$$

Inserting the solution into Eq. (18) and using the orthonormality of the basis matrices, gives the following expression for the squared norm,

$$\begin{aligned} R(\mu) &= \|D - \sum_{v=1}^{k} \hat{\alpha}_v^{\mu} A_v^{\mu}\|^2 \\ &= \left\langle D - \sum_{v=1}^{k} \hat{\alpha}_v^{\mu} A_v^{\mu},\ D - \sum_{v=1}^{k} \hat{\alpha}_v^{\mu} A_v^{\mu} \right\rangle \\ &= \langle D, D \rangle - \sum_{v=1}^{k} \langle D, A_v^{\mu} \rangle^2 = 1 - \sum_{v=1}^{k} \langle D, A_v^{\mu} \rangle^2. \end{aligned} \tag{20}$$

We now ascribe D to the class, for which $R(\mu)$ is smallest.

The classification algorithm is summarized in Algorithm 1 and test results are given in Section 4.3.

Algorithm 1. Classification by HOSVD

- Training phase:
  - (1) Sort the training digits into tensors with digits of the same type.
  - (2) Compute the HOSVD of the tensors.
  - (3) Compute and store the normalized basis matrices $(A_v^{\mu})_{v=1}^{k}$, $\mu = 0, 1, \ldots, 9$.
- Test phase:
  - (1) Normalize the unknown digit.
  - (2) Compute $R(\mu) = 1 - \sum_{v=1}^{k} \langle D, A_v^{\mu} \rangle^2, \mu = 0, 1, \ldots, 9$.
  - (3) Determine $\mu_{\min} := \operatorname{argmin}_{\mu} R(\mu)$ and classify D as $\mu_{\min}$.

[5] The digits in their original form would give vectors in $\mathbb{R}^{16\times 16}$ but the preprocessing with the blurring operator enlarges the digit size to 20×20 pixels. See Section 4.2 for details.

[6] In the SIMCA algorithm, usually the SVD is used. By reordering the digits as vectors, we could have done this also in this application.

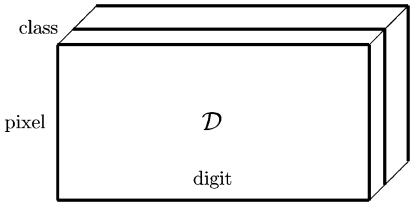

Fig. 6. The tensor with all the digits. The dimensions are 400 for the pixel mode, 10 for the class mode and approximately 1000 for the digit mode.

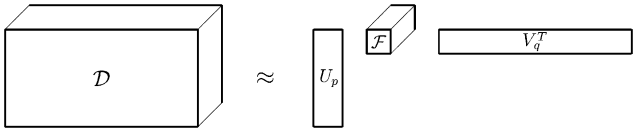

Fig. 7. Illustration of the tensor approximation.

3.2. Algorithm 2: Compression and classification by HOSVD

HOSVD can be used to compress the training set prior the computation of the basis vectors for the different classes. This is the idea behind the second algorithm and the main contribution of this paper. There is a close analogy with low rank matrix approximations.

The advantage in terms of computational efficiency is that the digits from different classes are all projected onto one common subspace. Therefore, an unknown digit is only projected once. If there were different subspaces for the different classes we would have to project an unknown digit on each one of these subspaces. The test phase of the algorithm would require a considerably larger amount of computations and memory.

3.2.1. Training phase

First we build a tensor with all the digits in the training set. The data can be visualized as in Figure 6. All the digits are reshaped into vectors in $\mathbb{R}^{400}$. They are sorted so that every slice contains digits for one class only.

Let $\mathcal{D}$ have the HOSVD

$$\mathcal{D} = \mathcal{S} \times_1 U \times_2 V \times_3 W \approx \mathcal{F} \times_1 U_p \times_2 V_q, \tag{21}$$

where $U_p = U(:, 1:p)$, $V_q = V(:, 1:q)$ and $\mathcal{F} = \mathcal{S}(1:p, 1:q, :) \times_3 W$. Figure 7 illustrates the tensor approximation.

By this approximation we reduce the representation of the individual digits from $\mathbb{R}^{400}$ to $\mathbb{R}^p$ and the amount of digits in each class to q. The reduced tensor $\mathcal{F}$ can be computed as

$$\mathbb{R}^{p \times q \times 10} \ni \mathcal{F} = \mathcal{D} \times_1 U_p^T \times_2 V_q^T. \tag{22}$$

We can view the reduced image representation as a projection onto the column-space of U_p.

Under the assumption that both p and q are much smaller than the corresponding dimensions of the tensor $\mathcal{D}$ we can achieve a considerable reduction of the original training set.

Table 1
Relative error (in norm) of the tensor approximation and (data reduction) in % for a few various p and q

$p \backslash q$	32	48	64
32	9.27 (99.45)	7.11 (99.18)	5.70 (98.90)
48	9.10 (99.18)	6.86 (98.77)	5.39 (98.35)
64	9.07 (98.90)	6.82 (98.35)	5.33 (97.81)

Table 1 gives the relative error in the approximation and the corresponding data reduction for various p and q. In order for the approximation to be good it is necessary that the omitted pixel and digit mode singular values are relatively small. Part of these singular values are shown in Figure 8. We see that the pixel mode singular values, corresponding to the representation of the digits, decay rather fast. The decay is not that fast for the digit mode singular values indicating the larger variation of the different digits of all the classes. But still, for the p and q in Table 1, the error rates[7] are surprisingly low, even for the cases where the data reduction is more than 99%.

Rewriting the approximation from Eq. (21) as a low dimensional representation of the digit tensor in the pixel mode

$$\mathcal{D}_p = \mathcal{D} \times_1 U_p^T = \mathcal{F} \times_2 V_q, \tag{23}$$

gives the following interpretation of the HOSVD decomposition. Each column vector in $\mathcal{D}_p$ is a p element representation of some digit. The different slices of $\mathcal{F}$ on the right-hand side consist of q basis vectors for the different classes.

Rows of V_q are the coordinates in terms of the basis vectors for various digits. The interesting part is that one row of V_q gives the coordinates for digits belonging to all 10 classes. See Figure 9 for a visualization.

The columns of $F^\mu := \mathcal{F}(:, :, \mu)$ represent basis vectors for some class given by μ. To obtain orthonormal and ordered basis vectors for the different classes we compute the SVD of the F^μ and take the k most significant left singular vectors,

$$F^\mu = [B^\mu (B^\mu)^\perp] \Sigma^\mu (Q^\mu)^T, \quad \mu = 0, 1, \ldots, 9,$$

where the k-column basis matrices we have been denoted $B^\mu \in \mathbb{R}^{p \times k}$. Note also that columns of B^μ span the dominant k-dimensional subspace of F^μ.

3.2.2. Test phase

Let $d \in \mathbb{R}^{400}$ be an unclassified digit. In the test phase we compute the low dimensional representation $d_p = U_p^T d$ and solve the following much smaller least squares problems,

$$\min_{x^\mu} \|d_p - B^\mu x^\mu\|, \quad \mu \text{ fixed class index.} \tag{24}$$

[7] Error rates for the algorithm are presented in Section 4.

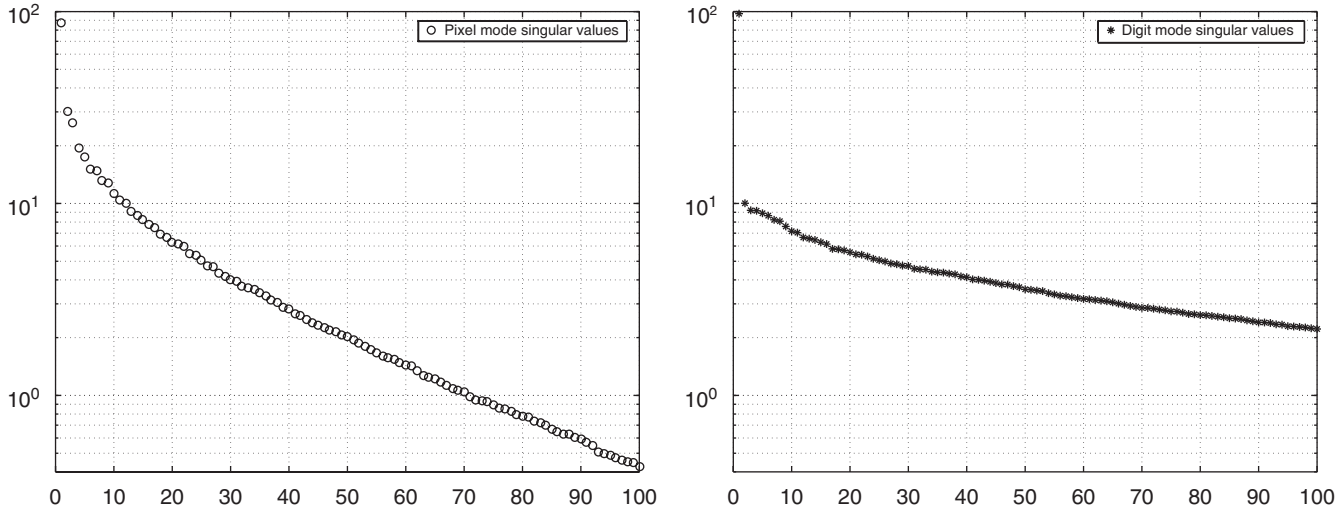

Fig. 8. Singular values for the pixel mode (left) and for the digit mode (right) of the tensor $\mathscr{D}$.

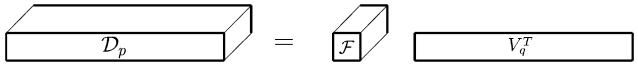

Fig. 9. Reduction of the digits in the pixel mode as a *class independent coordinate decomposition*.

Since the columns of B^μ are orthonormal the solution is given by

$$\hat{x}^\mu = (B^\mu)^T d_p.$$

Inserting $\hat{x}^\mu$ into Eq. (24) gives

$$R(\mu) = \|d_p - B^\mu (B^\mu)^T d_p\|.$$

Once again the index μ that gives the smallest residual also gives the predicted class of the unknown digit.

The complete classification algorithm is as follows.

Algorithm 2

- Training phase:

(1) Sort and vectorize the digits from the training set into a tensors $\mathscr{D}$.
(2) Compute the HOSVD of $\mathscr{D}$, Eq. (21).
(3) Compute the reduced tensor $\mathscr{F}$ representing the training set, Eq. (22).
(4) Compute and store the basis matrices B^μ for each class.

- Test phase:

(1) Compute the low dimensional representation $d_p = U_p^T d$ of an unknown digit.
(2) Compute the residuals $R(\mu) = \|d_p - B^\mu (B^\mu)^T d_p\|$, $\mu = 0, 1, \ldots, 9$.
(3) Determine $\mu_{\min} = \text{argmin}_\mu R(\mu)$ and classify d as $\mu_{\min}$.

4. Tests and results

The process described this far is general and there are several parameters that can be varied to customize an algorithm. One such parameter is the number k of basis matrices to be used in the classification. Other parameters are p and q in the second algorithm. In this section we present the tests performed to validate the algorithms. But first we give a brief description of the data set we used in the tests and how it was preprocessed.

4.1. The data set—US postal service database

The data set we use in our experiments is freely available on the Internet[8] and is frequently used for evaluation of classification algorithms. The digits in the database are extracted by scanning the ZIP code on envelopes from US Postal mail. The size of the images is 16×16 pixels and each pixel has a gray scale intensity range of 0–255.

Two sets are available, a training set containing 7291 digits and a test set containing 2007 digits. The digit distribution is given in Table 2.

According to Hastie [2] this set is rather difficult from a classification point of view when compared to other data sets. Note that the images in Figure 1 are quite well written, but there are many others that are badly written.

The MNIST database is another well-known handwritten digit set used for classification purpose. Here the digit size is 28×28 pixels with the same intensity range. The digits are rather well written and in general algorithms give considerably lower error rate for this database [5,20].

In order to use the complete training set in the 3-tensors of the algorithms some digits were duplicated. The reason

[8] The database is available in the web-page of Hastie [2].

Table 2
The digit distribution in the US Postal Service data sets

	0	1	2	3	4	5	6	7	8	9	Total
Train	1194	1005	731	658	652	556	664	645	542	644	7291
Test	359	264	198	166	200	160	170	147	166	177	2007

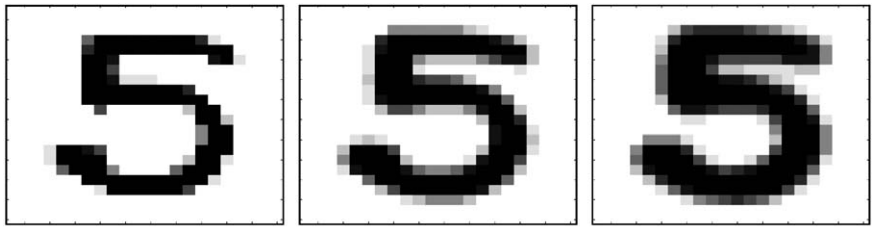

Fig. 10. Examples of the effect of the Gaussian blurring operator: (a) original; (b) blurred, $\sigma = 0.7$; (c) blurred, $\sigma = 1$.

for this is the different amount of digits for the different classes.

4.2. Preprocessing—blurring with a Gaussian

The classification data can be preprocessed in several ways [21,1]. Some kind of blurring and normalization are usual preprocessing techniques. According to Simard [5] blurring is of great importance for the identification process, at least when classifying handwritten digits. The blurring can be described as smoothing the pattern or making sharp edges and corners softer.

Different kinds of blurring can be obtained depending on the function used in the operation. One usual function is the Gaussian,

$$g(x, y) = \mathrm{e}^{-(x^2+y^2)/(2\sigma^2)}. \tag{25}$$

The standard deviation σ is used to control the amount of blurring. Figure 10 gives two examples.

Due to the fast decay of Eq. (25), we approximate the pure Gaussian blurring by restricting its influence only to the immediate neighborhood of the corresponding pixels, as indicated in Figure 11. The area with the darker squares is the neighborhood the Gaussian is allowed to influence. This approximation of the Gaussian blurring gives considerable reduction of the computation time. In addition the approximate blurring is almost indistinguishable compared with the complete blurring. The blurring factor is set to $\sigma = 0.9$, the same as used in Ref. [5]. This σ value is considered to give high enough blurring without reducing the information contents in the patterns.

4.3. Tests and results for Algorithm 1

Tests were conducted according to the algorithm formulation in Section 3.1. Up to 16 basis matrices for each class were used in the classification and 10 least squares problems were solved for every unknown. In each test the number of basis matrices was the same for all the classes. The test results are shown in Figure 12. Clearly, the algorithm performs better when the number of basis matrices is increased up till 12. This does not mean that the incorrect classifications for all of the different classes decrease, but the overall result gets better. The incorrect classifications within the different classes is not uniformly distributed. Two's, three's, four's, five's and eight's are generally considerably harder to classify.

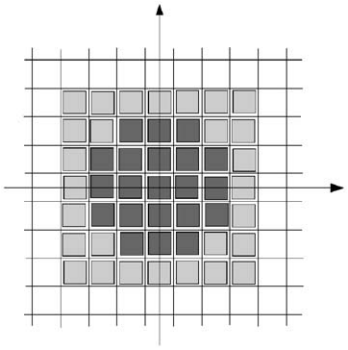

Fig. 11. The Gaussian influences only the immediate neighborhood, the area with darker squares, of the pixel at the origin.

Sometimes it is important to be confident that an algorithm makes the right classification in every case. If there is a high risk that an incorrect classification is to be made then it might be better to sort out the object in question for further analysis. Such a property is easily incorporated by replacing the third step in the test phase of the algorithm with the following;

3. If the lowest residual is significantly lower then all the other residuals then take the label of that basis set, else reject the digit as unidentified.

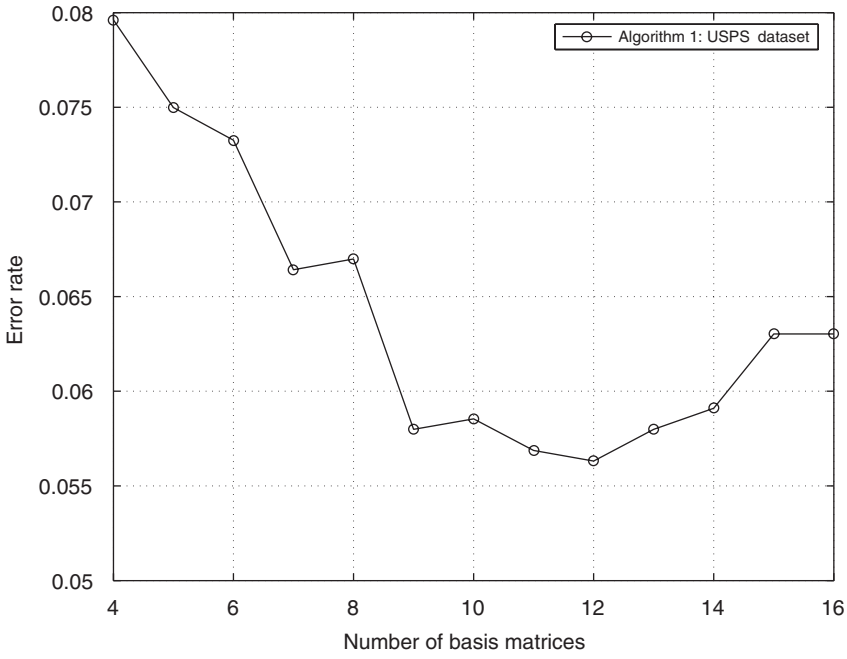

Fig. 12. Classification results (error rate) for the USPS data set.

4.4. Tests and results for Algorithm 2

The tests for the second algorithm were performed in a similar manner. We varied the number of basis vectors and we also varied p and q, which determine the amount of data reduction, according to the values given in Table 1. The results are presented in Figure 13. The classification results for the second algorithm are rather remarkable. Even though we have made data reductions by approximately 98–99% the algorithm performs almost as well as Algorithm 1, which used the complete training set in the model construction. In some cases the error is even smaller than the error rate of Algorithm 1, compare Figures 12 and 13.

4.5. Computational complexity

How fast an algorithm performs the classification is often of great importance. The training phase is also important but not that crucial for the real time performance of an application. Here we give an overview of the computational complexity for the presented algorithms and compare it with a nearest neighbor algorithm.[9] In the following we make the assumptions that the size of the digits are $I \times I$ pixels, and each class has N training digits and that k basis vectors (matrices) are used in the classification.

4.5.1. Training phase

In the training phase of Algorithms 1 and 2 we compute A_v^μ and B^μ, respectively. To obtain A_v^μ we need to compute W^μ in $\mathcal{A}^\mu = S^\mu \times_1 U^\mu \times_2 V^\mu \times_3 W^\mu$ and then perform $\mathcal{A}^\mu \times_3 (W^\mu)^T$. These operations require an SVD computation and a tensor–matrix product. In total we need to do $60N^2I^2 + 220I^6$ floating point operations (flops). The dominating part of the computations in Algorithm 2 is to calculate U_p and V_q in Eq. (21) which involves the SVDs of two possibly rather large matrices. For this we need to do approximately $22000(N^3 + I^6)$ flops.

4.5.2. Test phase

The test phase of Algorithm 1 consists mainly of the computation of $\langle D, A_v^\mu \rangle$ for $\mu = 0, 1, \ldots, 9$ and $v = 1, \ldots, k$ yielding a total of $10k$ scalar products.[10] In Algorithm 2 we perform $d_p = U_p^T \mathrm{vec(D)}$ and $\mathrm{norm}((I - B^\mu (B^\mu)^T) d_p)$ yielding approximately $p(1 + 40k/I^2)$ scalar products. A nearest neighbor algorithm involves a comparison of an unknown digit with all digits in the training set resulting in $15N$ scalar products.

4.5.3. Test phase memory requirements

The memory requirements for the algorithms is essentially the storage of the basis vectors for the different classes. In Algorithm 1 we need to access $10k$ basis vectors in $\mathbb{R}^{I \times I}$. Algorithm 2 needs also $10k$ vectors but in $\mathbb{R}^p$. The nearest neighbor algorithm stores the complete training set. An example of the computational complexity and memory requirements for the three algorithms is given in Table 3.

[9] The nearest neighbor algorithm on the USPS data set has an error rate of approximately 8.5%.

[10] The scalar products are on vectors in $\mathbb{R}^{I^2}$, μ is the class index and v indicates the different basis matrices.

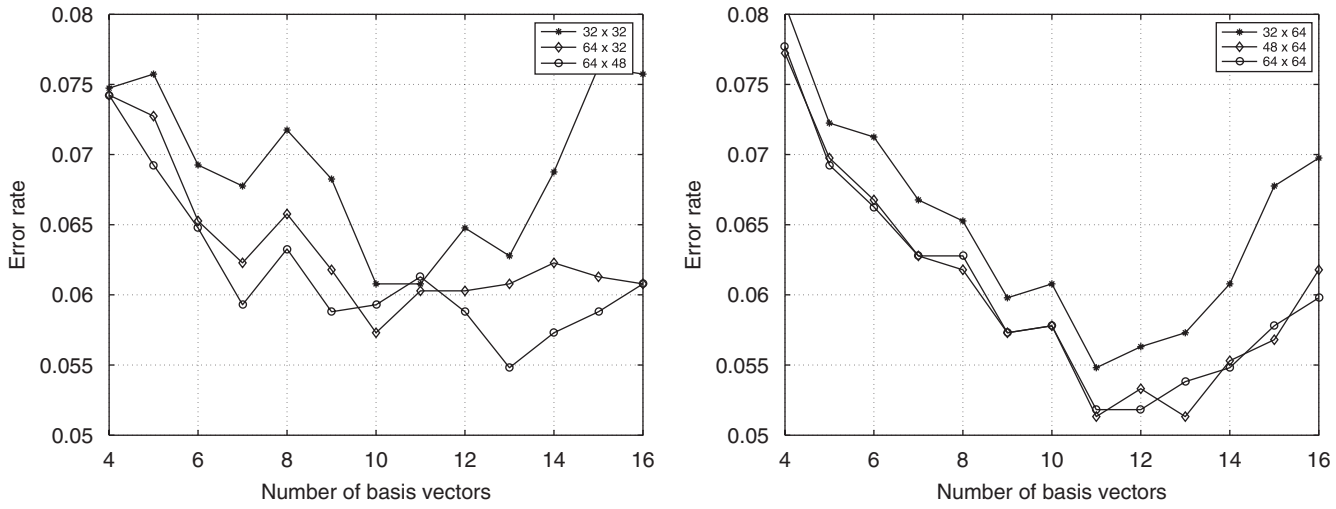

Fig. 13. Classification results for the second algorithm with the reduction parameters $p \times q$ given in the legend window in each figure.

Table 3
Computational complexity and memory requirements for a specific case: $N = 1000$, $I = 20$, $k = 10$ and $p = 32$

	Algorithm 1	Algorithm 2	Nearest neighbor
Training (flops)	38×10^9	23×10^{12}	0
Test (scalar products)	100	64	15 000
Memory (in digit size)	100	8	10 000

4.6. Future work

Analyzing the distributions of the α_v^μ coefficients in Eq. (18) might give further insight and perhaps improve the performance of the algorithm. A quick check of this revealed that different digits have different coefficient distributions. This may also be useful for detecting less well written digits.

One can also analyze the subspaces spanned by the basis vectors describing the different classes. Computation of the principal angles or distance between the subspaces for the different classes might also give some insight on the description and relative position of the different kinds of digits in space. Some analysis in this area is found in [20].

Presently we are interesting how to incorporate the tangent distance [5] in the second algorithm. The tangent distance in its pure form is very computationally heavy but gives very good classification results. Since the training set is reduced in both the pixel mode (low dimensional digit representation) and the digit mode (each class is described with a few basis) a hybrid version has the potential to be fairly effective without considerable loss in classification.

5. Conclusions

In this paper we have presented two rather simple and computationally effective algorithms based on linear and multilinear theory. We have showed that both algorithms achieve satisfactory classification results. Particulary the main algorithm in the paper gives an error rate of 5–5.5% even after reducing the training set with more than 98% prior the construction of the class specific models.

References

[1] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, second ed., Wiley-Interscience, New York, 2001.
[2] T. Hastie, R. Tibshirani, J. Friedman, The elements of statistical learning, Springer Series in Statistics, Springer, New York, 2001.
[3] Y. LeCun, L. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Müller, E. Säckinger, P. Simard, V. Vapnik, Learning algorithms for classification: a comparison on handwritten digit recognition, Neural Network: The Statistical Mechanics Perspective (1995) 261–276.
[4] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[5] P.Y. Simard, Y.A. LeCun, J.S. Denker, B. Victorri, Transformation invariance in pattern recognition—tangent distance and tangent propagation, Internat. J. Imag. Systems Technol. 11 (3) (2000) 181–197.
[6] D. Keysers, J. Dahmen, T. Theiner, H. Ney, Experiments with an extended tangent distance, Technical Report, Lehrstuhl für Informatik VI, RWTH Aachen - University of Technology, 52056 Aachen Germany, 2000.
[7] P. Scattolin, Recognition of handwritten numerals using elastic matching, Master's Thesis, Department of Computer Science, Concordia University, Montréal, Québeck, Canada, 1995.
[8] G.E. Hinton, P. Dayan, M. Revow, Modelling the manifolds of images of handwritten digits, IEEE Trans. Neural Networks 8 (1) (1997) 65–74.
[9] L. De Lathauwer, B. De Moor, J. Vandewalle, A multilinear singular value decomposition, SIAM J. Matrix Anal. Appl. 21 (4) (2000) 1253–1278.
[10] S. Wold, Pattern recognition by means of disjoint principal components models, Pattern Recognition 8 (1975) 127–139.
[11] A. Smilde, R. Bro, P. Geladi, Multi-way Analysis: Applications in the Chemical Sciences, Wiley, New York, 2004.

[12] M. Vasilescu, D. Terzopoulos, Multilinear subspace analysis of image ensembles, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03), Madison WI, 2003.
[13] M. Vasilescu, D. Terzopoulos, Multilinear analysis of image ensembles: tensorfaces, in: Proceedings of Seventh European Conference on Computer Vision (ECCV'02), Lecture Notes in Computer Science, vol. 2350, Springer, Copenhagen, Denmark, 2002, pp. 447–460.
[14] B. Bader, T. Kolda, Matlab tensor classes for fast algorithm prototyping, Tech. Rep. SAND2004-5187, Sandia National Laboratories (October 2004).
[15] S. Kobayashi, K. Nomizu, Foundations of Differential Geometry, vol. 1, Wiley, New York, 1963.
[16] G.H. Golub, C.F. Van Loan, Matrix Computations, third ed., Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 1996.
[17] L. Tucker, The extension of factor analysis to three-dimensional matrices, in: H. Gulliksen, N. Frederiksen (Eds.), Contributions to Mathematical Psychology, Holt, Rinehart and Winston, New York, 1964, pp. 109–127.
[18] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (1936) 211–218.
[19] L. De Lathauwer, B. De Moor, J. Vendewalle, On the best rank-1 and rank-$(r_1, r_2, \ldots, r_n)$ approximation of higher-order tensors, SIAM Matrix Anal. Appl. 21 (4) (2000) 1324–1342.
[20] B. Savas, Analyses and tests of handwritten digit recognition algorithms, Master's Thesis, Linköping University, ⟨http://www.mai.liu.se/∼besav/⟩ (2003).
[21] S. Mori, H. Nishida, H. Yamada, Optical Character Recognition, Wiley, New York, 1999.