Principal Component Analysis of symmetric fuzzy data

Paolo Giordani\textsuperscript{a,}\textsuperscript{*}, Henk A.L. Kiers\textsuperscript{b}

\textsuperscript{a}Department of Statistics, Probability and Applied Statistics, University of Rome “La Sapienza”, P.le Aldo Moro, 5, I-00185 Rome, Italy
\textsuperscript{b}Heymans Institute (DPMG), University of Groningen, Grote Kruisstraat 211, 9712 TS Groningen, The Netherlands

Accepted 2 December 2002

Abstract

Principal Component Analysis (PCA) is a well-known tool often used for the exploratory analysis of a numerical data set. Here an extension of classical PCA is proposed, which deals with fuzzy data (in short PCAF), where the elementary datum cannot be recognized exactly by a specific number but by a center, two spread measures and a membership function. Specifically, two different PCAF methods, associated with different hypotheses of interrelation between parts of the solution, are proposed. In the first method, called Centers-related Spread PCAF (CS-PCAF), the size of the spread measures depends on the size of the centers. In the second method, called Loadings-related Spread PCAF (LS-PCAF), the spreads are not related directly to the sizes of the centers, but indirectly, via the component loadings. To analyze how well PCAF works a simulation study was carried out. On the whole, the PCAF method performed better than or equally well as PCA, except in a few particular conditions. Finally, the application of PCAF to an empirical fuzzy data set is described.

© 2002 Elsevier B.V. All rights reserved.

Keywords: Principal Component Analysis; Fuzzy data sets; Least squares approach

1. Introduction

There are many practical situations in which numerical observations of \(I\) units with respect to \(J\) variables cannot be represented by precisely specified values. Examples can be found in economics (daily rate exchanges), in marketing research (assessing
public opinions about special events like government elections, launching of new consumer goods), in psychology (the mental state or subjective perceptions), in physics (in a ballistic sphere, the needed velocity of perforation of a plate by projectiles). Another significant situation is related to the attempt to analyze subjective perception or linguistic variables. In each of the above examples an exact numerical coding seems to be very hard to give: thus, the available information can only be represented approximately.

In the above situations the information is vague or fuzzy and can be summarized using interval valued data instead of numerical data. Thus, the score of the generic observation \( i \) on the generic variable \( j \) is represented by a couple of values: a lower bound and an upper bound which enclose the exact observation. In contrast to the interval valued data framework, the fuzzy data approach offers the possibility to take into account additional information given by the membership function. This function assigns a different “role” to each value of an interval in contrast to the interval valued data approach in which each value has a uniform importance. This distinction will be clearer after defining a fuzzy number, as follows.

A fuzzy number is defined as the triple \( F = (m, l, r)_{LR} \) where \( m \) denotes the center and \( l \) and \( r \) are the left and right spread, respectively, with the following membership function

\[
\mu_F(x) = \begin{cases} 
L \left( \frac{m - x}{l} \right), & x \leq m \ (l > 0), \\
R \left( \frac{x - m}{r} \right), & x \geq m \ (r > 0), 
\end{cases}
\] (1)

where \( L \) and \( R \) are continuous strictly decreasing functions on \([0, 1]\) called shape functions. Moreover these functions must fulfil additional requirements: for example with reference to \( L \), \( L(0) = 1 \), \( L(z) < 1 \) if \( z > 0 \), \( L(z) > 0 \) if \( z < 1 \) and \( L(1) = 0 \). Further details can be found in Dubois and Prade (1980).

Hereafter, for the sake of simplicity, we consider ‘symmetric’ fuzzy numbers only. For a symmetric fuzzy number \( l = r \) and \( L = R \). It follows that a symmetric fuzzy number can be completely identified by the couple \( F = (m, s)_{LR} \) where \( s = l = r \) is the spread. Thus, a fuzzy data set is a set of fuzzy numbers, which represent the scores of \( I \) observation units on \( J \) symmetric fuzzy variables. By a symmetric fuzzy variable we mean a variable for which each observed datum cannot be quantified exactly but only by using the couple \( F = (m, s)_{LR} \).

In this paper we generalize classical Principal Component Analysis (PCA) to deal with fuzzy data sets using a least squares approach. A few proposals for this are available in the literature: Yabuuchi et al. (1997) propose a method to perform PCA on fuzzy data, in which fuzzy eigenvalues and crisp (non-fuzzy) eigenvectors are obtained by solving a linear programming problem. Cazes et al. (1997) and extensions of their approach (see, for example, Palumbo and Lauro, 2002) propose to carry out factorial decompositions of interval valued data as well as a probabilistic generalization thereof. The basic idea consists of performing PCA on the bounds or on the centers of the interval valued data. In their probabilistic generalization, they make use of a coefficient the value of which depends on the probability law at hand that modifies the information taken into account in the factorial decomposition.
The need for generalizing PCA to fuzzy data is based on the assumption that, as well as for single valued data, it can be helpful to synthesize the data without losing relevant information. In the next section, we present a general least squares approach to fuzzy data analysis. In Section 3, we propose two different models that extend PCA to fuzzy data and in Section 4 an alternating least squares algorithm to estimate the solutions will be given. In Section 5, we discuss the representation of each observation unit in the low dimensional subspace obtained by PCAF. In Sections 6 and 7, the results of a simulation study to compare the PCAF to classical PCA and an application of PCAF to real fuzzy data will be given.

2. Least squares approach to fuzzy data analysis

The need of studying and understanding fuzzy data has led to a growing interest in fuzzy data analysis. Many authors have dealt with fuzzy regression analysis in the past as well as nowadays. In the least squares sense, the analysis involves a minimization problem of a distance function between two sets of values, the empirical data set and the values estimated according to the specific model involved. In this framework we refer to Diamond (1988) who has developed a fuzzy least squares method based on a compatibility measure between two fuzzy numbers, the observed one and the estimated one. D'Urso and Gastaldi (2000) have proposed a fuzzy regression problem based on the minimization of a new distance expressing the sum of squared estimation errors of the centers and the spreads. Therefore, we have

$$L = R/S_{OH}^2 = \|m - m^*\|^2 + \|s - s^*\|^2,$$

where $m$ and $m^*$ are the vectors, respectively, of observed and estimated centers of the fuzzy dependent variable and $s$ and $s^*$ are the vectors, respectively, of observed and estimated spreads of the fuzzy dependent variable. The estimation vectors are obtained by using two different regression models that relate the dependent fuzzy variable to a set of independent crisp variables.

A possible drawback of (2) is that the vectors of centers and spreads have the same weight in the estimation procedure. Coppi and D’Urso (2001) proposed a more flexible approach, which takes into account information due to spreads but, at the same time, focuses especially on the centers information in the estimation procedure, because in these points the membership function value is maximal. Specifically, they proposed to take into account the specific membership function of the fuzzy variable in order to weight in the right way the role of the spreads, by using the distance:

$$L = R/A_{\lambda}^2 = \|m - m^*\|^2 + \|(m - \lambda s) - (m^* - \lambda s^*)\|^2 + \|(m + \lambda s) - (m^* + \lambda s^*)\|^2,$$

where $m$, $m^*$, $s$ and $s^*$ are the vectors defined above, while, unlike in (2), in (3) the parameter $\lambda = \int_{0}^{1} L^{-1}(\omega) d\omega$ appears. Based on the idea by Yang and Ko (1996), it has a twofold role: taking into account the variability of the membership function and decreasing the emphasis on the estimation of spreads. To explain this, we calculate
the parameter \( \lambda \) for a particular membership function, namely, for the very common triangular membership function

\[
\mu_T(x) = \begin{cases} 
1 - \frac{|m - x|}{s} & \text{if } m - s \leq x \leq m + s, \\
0 & \text{otherwise.}
\end{cases}
\]  

(4)

Let \( x = (m - x)/s \), it follows that

\[
L(x) = \begin{cases} 
1 - x & \text{if } 0 \leq x \leq 1, \\
0 & \text{otherwise,}
\end{cases}
\]  

(5)

and therefore

\[
L^{-1}(\omega) = \begin{cases} 
1 - \omega & \text{if } 0 \leq \omega \leq 1, \\
0 & \text{otherwise.}
\end{cases}
\]  

(6)

From (6) we have

\[
\lambda = \int_0^1 L^{-1}(\omega) \, d\omega = \int_0^1 (1 - \omega) \, d\omega = 1/2.
\]  

(7)

The parameter \( \lambda \) is often a number less than one except in the unlikely situation where the importance of the points increases as they are farther from the center. However, for the most common membership functions (normal, trapezoidal) the parameter \( \lambda \) is always lower than one but, at the same time, depends on the membership function at hand. Thus, in the distance in (3), it reduces the emphasis of the spreads in a different way based upon the membership function at hand. If the membership function values are high only very near to the center, it follows that the parameter \( \lambda \) is very small because the surface under \( L^{-1}(\omega) \) is small. On the contrary, if the membership function values are high in almost all the interval \([m - s, m + s]\) and decrease very near to the end points of the interval it follows that the surface under \( L^{-1}(\omega) \) is very big and therefore \( \lambda \) assumes a bigger value. See, for more details, Yang and Ko (1996).

Following the idea of Yang and Ko (1996), we calculate the integral with respect to the variable \( \omega \), which takes into account the fuzziness of the data. At the same time, we suggest to avoid using the inverse of the membership function. Bearing in mind the requirements of the membership function it follows that

\[
\int_0^1 L(x) \, dx = \int_0^1 L^{-1}(\omega) \, d\omega = \lambda,
\]  

(8)

because we deal with exactly the same surface by mirroring the graph of \( L(x) \) against \( x \) to obtain that of \( L^{-1}(\omega) \) against \( \omega \). We think that the first term in (8) is an easier and more comprehensible expression to define the parameter \( \lambda \).

3. Principal Component Analysis of symmetric fuzzy data (PCAF)

In this paper we propose a principal component model for fuzzy data by extending (3) to deal with matrices instead of vectors. If each observation unit is represented by
a score on a single fuzzy variable, the information can be represented as a segment in \( \mathbb{R}^1 \). It follows that, if one wants to compare two units, it is sufficient to compare the center and the two vertices as in (3). Instead, if two different variables are associated with each unit, a generic unit is represented as a rectangle in \( \mathbb{R}^2 \). Obviously the number of vertices is \( 2^2 = 4 \). In the general case of \( J \) variables, each of the \( I \) units is represented by a hypercube in \( \mathbb{R}^J \) and the total number of vertices is \( 2^J = K \). Differently from the interval valued data, in which points lie uniformly distributed in that hypercube, with fuzzy data the distribution of the points depends on the \( J \) membership functions.

To extend (3) we have to take into account the differences between the observed and estimated centers and the differences between every observed and estimated vertex. In fact, we do not use exactly the vertices but a sort of scaled version taking into account the membership function (as expressed by the parameter \( \lambda \)). This idea is justified by noting that in each vertex the membership function value is equal to 0. Thus we suggest to use, for each vertex, a sort of ‘middle’ point between the center and the vertex, the position of which depends on the particular membership function, hence on the particular value of the parameter \( \lambda \). In analogy to (3), but now taking into account all the vertices of the hypercube associated with each unit, we get

\[
L_{\hat{R}^2} A^2 = \| M - M^* \|^2 + \sum_{k=1}^{K} \| (M + S \hat{H}_k A) - (M^* + S^* \hat{H}_k A) \|^2,
\]

where \( M, M^* \), \( S \) and \( S^* \) are matrices with \( I \) rows and \( J \) columns of the observed centers and estimated ones and of the observed spreads and estimated ones, respectively. \( \hat{A} \) is a diagonal matrix of order \( J \), whose elements in the main diagonal are the parameters \( \hat{\lambda}_j \), \( j = 1, \ldots, J \). Therefore, we assume that every variable has its characteristic membership function which is the same for each unit. From (9) it should be noted that we assume that the estimated vertices are located symmetrically with respect to the associated estimated center. Moreover, we take the weight for the estimated spreads equal to those for the observed spreads. It remains to explain the matrices \( \hat{H}_k \), \( k = 1, \ldots, K \), where \( K = 2^J \). Their role is to help defining every vertex separately. They allow us to consider all the vertices of the hypercube associated with each unit. They are diagonal matrices whose elements in the main diagonal are \( \pm 1 \) in order to refer exactly to every vertex. A small example (\( J = 3 \)) can help to clarify these matrices. Let us introduce the following new matrix \( \hat{H}_{K \times J} \) of order \( K \times J \) (or \( 2^3 \times 3 \)):

\[
\hat{H}_{K \times J} = \begin{bmatrix}
-1 & -1 & -1 \\
-1 & -1 & 1 \\
-1 & 1 & -1 \\
1 & -1 & -1 \\
1 & 1 & 1 \\
1 & 1 & -1 \\
1 & -1 & 1 \\
-1 & 1 & 1
\end{bmatrix}.
\]

(10)
Their elements are $\pm 1$ but each row is related to a different vertex: for example the first row indicates the minimal vertex and the fifth row the maximal one. By minimal and maximal vertex, we mean, respectively, the vertices whose coordinates are the lower bounds and the upper bounds for all variables. The main diagonal elements of the generic matrix $H_k$ are exactly the elements of the $k$th rows of $H_{K \times J}$. Therefore, the matrices of the minimal and the maximal vertex are, respectively,

$$
M + SH_1 = M + S \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} m_{11} - s_{11} & m_{12} - s_{12} & m_{13} - s_{13} \\ \vdots & \vdots & \vdots \\ m_{I1} - s_{I1} & m_{I2} - s_{I2} & m_{I3} - s_{I3} \end{bmatrix} \quad (11)
$$

and

$$
M + SH_5 = M + S \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} m_{11} + s_{11} & m_{12} + s_{12} & m_{13} + s_{13} \\ \vdots & \vdots & \vdots \\ m_{I1} + s_{I1} & m_{I2} + s_{I2} & m_{I3} + s_{I3} \end{bmatrix} \quad (12)
$$

The usefulness will be clear later but it is very important to note that the rows of $H_{K \times J}$ are arranged in a specific way. It can easily be seen that $H_g + H_{2^j-1+g} = 0_j$, $g = 1, \ldots, 2^j-1$.

Now we will simplify expression (9). From (9) we consider the generic $k$th term of the sum. We have

$$
\|(M + SH_kA) - (M^* + S^*H_kA)\|^2 = tr(M'M) + tr(AH_kS'SH_kA) + 2tr(M'SH_kA) + tr(M^*M^*) \\
+ tr(AH_kS'S'SH_kA) + 2tr(M^*S'S'H_kA) - 2tr(M'M^*) \\
+ 2tr(M'S'H_kA) - 2tr(AH_kS'M^*) - 2tr(AH_kS'S'H_kA). \quad (13)
$$
We can see that we have expanded the generic \( k \)th term into nine traces. We can distinguish these nine terms in three different groups according to the matrix \( H_k \). In three traces \( H_k \) is absent, while it is present in the remaining ones. First we can consider the terms in which \( H_k \) is absent. We can easily see that these terms taken together give

\[
tr(M'M) + tr(M'S^*M) - 2tr(M'M^*) = ||M - M^*||^2.
\]

Let us now collect the four terms in (13) in which \( H_k \) appears just once:

\[
2tr(M'SH_kA) + 2tr(M'S^*H_kA) - 2tr(M'S^*H_kA) - 2tr(\Lambda H_kS'M^*)
\]

\[
= 2tr[(\Lambda M'S + \Lambda M'S^* - \Lambda M'S^* - S'M^*A)H_k].
\]

Using that \( H_g + H_{2^j-1+g} = 0 \), \( g = 1, \ldots, 2^{J-1} \), it is clear that

\[
2tr[(\Lambda M'S + \Lambda M'S^* - \Lambda M'S^* - S'M^*A)(H_g + H_{2^j-1+g})] = 0,
\]

for \( g = 1, \ldots, 2^{J-1} \). The result in (16) is very helpful in order to simplify (9), because, thus, all the terms in which \( H_k \) appears just once vanish.

Finally, we can analyze the three remaining traces in which \( H_k \) appears twice. We have

\[
tr(\Lambda H_kS'SH_kA) + tr(\Lambda H_kS'S^*H_kA) - 2tr(\Lambda H_kS'S^*H_kA)
\]

\[
= tr(\Lambda S'SA) + tr(\Lambda S'S^*A) - 2tr(\Lambda S'S^*A) = ||SA - S^*A||^2,
\]

where we use that the matrices \( H_k \) and \( \Lambda \) are diagonal and that \( H_kH_k = I_J, k = 1, \ldots, K \).

Using the relations (13)–(17), we can simplify (9) as

\[
L=R A^2_{\lambda} = (2^J + 1)||M - M^*||^2 + 2^J||SA - S^*A||^2.
\]

We think that (18) is a very useful way to express the differences between two \( I \times J \) matrices of fuzzy data where, in this context, the two matrices are the observed one and the estimated one, because we can use all the available information (centers, spreads and membership functions) and an objective system of weights to take into account the different kinds of information in the right way. In (9) we compared the centers and all the \( 2^J = K \) vertices (scaled down by \( \Lambda \)) of each hypercube associated with each observation unit. As a matter of fact, (9) can be very heavy to compute if the number of variables increases. In (18), (9) is simplified very much because we showed that instead of comparing all the centers and all the vertices (modified by \( \Lambda \)), we have to compare only the centers and the spreads (modified by \( \Lambda \)) and to take into account the system of weights. We note that (18) is the matrix generalization of formula (3.3) by Coppi and D’Urso (2001), which itself is the special case of (18) with \( J = 1 \):

\[
L=R A^2_{\lambda} = 3||\mathbf{m} - \mathbf{m}^*||^2 + 2\|\hat{\lambda}(\mathbf{s} - \mathbf{s}^*)\|^2.
\]

We have now introduced and discussed the objective function to minimize based on the squared distance between observed and estimated centers matrices and observed
and estimated spreads matrices. However, we still have not defined the model to be used in PCAF. In matrix notation the model in PCAF is

\[ M = M^* + E, \]

\[ M^* = A_M F', \]

\[ M + S H_k = M^* + S^* H_k + Z_k, \quad k = 1, \ldots, K; \]

where \( A_M \) and \( F \) are, respectively, the component scores matrix for the centers of order \( I \times P \) and the component loadings matrix of order \( J \times P \), where \( P \) is the number of components and \( E \) and \( Z_k \) are residual matrices of order \( I \times J \). It remains to introduce how we suggest to model the spreads matrix \( S^* \). Two different approaches will be discussed: Centers-related Spread PCAF (CS-PCAF) and Loadings-related Spread PCAF (LS-PCAF).

### 3.1. Centers-related spread PCAF (CS-PCAF)

The CS-PCAF method is related to the hypothesis that a linear relation exists between the fuzziness (the size of the spreads) and the magnitude of the centers, so we assume that such a relation holds for the estimated centers and the estimated spreads. The above hypothesis was first proposed by D’Urso and Gastaldi (2000). There are several real cases in which that hypothesis seems to be at least approximately correct in different fields: for instance car speed, astronomical distance, income, rate of exchange.

Thus, in CS-PCAF we have the following expression for the estimates of the spreads

\[ S^* = M^* B + 1_{I \times J} D, \]

where \( 1_{I \times J} \) is a matrix of order \( I \times J \) with unit elements and \( B \) and \( D \) are diagonal matrices of order \( J \) whose main diagonal elements, respectively, are \( b_j \) and \( d_j \). Thus, we assume that there is a linear relation between centers and spread which is different for each variable but is the same for each observation unit. Substituting (20b) in (21) we find

\[ S^* = A_M F' B + 1_{I \times J} D \]

which displays how the estimated spreads are related to the PCA estimates of the centers.

Fitting the CS-PCAF model consists of minimizing (18), where \( M^* \) and \( S^* \) are obtained from (20b) and (21), with respect to \( A_M, F, B, \) and \( D \). As we will see below, upon substituting (21) in (18), it is clear that the information of the spreads affects the optimal estimates of the matrices of component scores and component loadings of the centers.

### 3.2. Loadings-related spread PCAF (LS-PCAF)

The CS-PCAF model in (20a)–(20c) and (21) is a parsimonious model for PCAF. However the hypothesis between centers and spreads is not always a reasonable assumption. In many situations, it seems quite hard to prove its validity. For that reason
we also propose a second method, LS-PCAF, which explains the phenomenon using more parameters, giving a more general model. In that situation we suggest the following relation to estimate the spreads matrix

\[ S^* = A_S F', \]  

(23)

where \( F \) is the component loadings matrix defined in (20) and \( A_S \) is the component scores matrix for the spreads. Thus, we assume that the centers and the spreads use the same component loadings \( F \). In this case, comparing (20b) and (23), the role of the spreads in the estimation procedure becomes clear. In fact, the component loadings matrix \( F \) is obtained taking into account the centers and the spreads information although making use of different weights.

At this point, the question is whether or not it is possible to find the same components for the centers and for the spreads. In fact, it is a trivial question: “it is always possible to obtain the same components in all groups, by simply defining them in the same way in all groups, albeit that they need not account for much variance in all groups. Hence the question whether or not it is possible to find the same components in different groups is meaningless” (Kiers and Ten Berge, 1994). It follows that the main question is whether such components are able to represent the data in each group sufficiently well, so it is important with this method to verify the goodness of fit of both the centers and the spreads.

A simple situation in which the LS-PCAF model seems to be adequate is the following one pertaining to the fuzzy scores of six observation units on four variables. We have the centers matrix

\[
M = \begin{bmatrix}
73.3 & 73.7 & 30.4 & 29.9 \\
46.7 & 46.8 & 24.1 & 24.6 \\
69.8 & 70.0 & 32.2 & 32.5 \\
74.1 & 74.4 & 31.4 & 31.6 \\
52.0 & 52.3 & 45.3 & 45.7 \\
66.6 & 66.8 & 28.7 & 28.4
\end{bmatrix}
\]  

(24)

and the spreads matrix

\[
S = \begin{bmatrix}
3.1 & 3.2 & 8.5 & 8.6 \\
5.5 & 5.3 & 9.9 & 10.0 \\
5.9 & 5.7 & 10.8 & 10.4 \\
2.9 & 3.0 & 8.7 & 8.5 \\
4.3 & 4.4 & 7.5 & 7.7 \\
2.1 & 2.3 & 9.3 & 9.4
\end{bmatrix}
\]  

(25)

Now, clearly, the underlying factor structure in (24) and (25) has two components related, respectively, to the first two variables and to the last two ones. Obviously, in
the above example, the LS-PCAF model deals with the data set very well. In other situations we think that the model recovers a sort of ‘compromise’ structure between centers and spreads taking into account the centers structure more than the spreads structure due to the different weights and to the parameters in matrix $A$.

To fit the LS-PCAF model, we minimize (18), in which the matrices of the estimated centers and of the estimated spreads are obtained, respectively, in (20b) and (23), with respect to $A_M$, $A_S$ and $F$.

3.3. Preprocessing step

Before performing PCAF, it is very useful to preprocess the raw data. We suggest to use different procedures for centers and for spreads. First of all, we standardize the centers matrix using the mean and the standard deviation of the centers values of each variable. After that we divide the spreads of each variable by the standard deviation of the centers of the variable at hand. In this way we eliminate artificial range differences but we keep the information about the sizes of the spreads.

To explain how the existing linear relation between centers and spreads in the CS-PCAF model is modified by using different preprocessing procedures, we first introduce the centering operator $L = I - I_1 I_1'$. The spreads are related to the centers but not to the centered centers. It follows that, in the CS-PCAF model, upon centering the centers, (20a) and (20b) must be replaced by

$$LM = LM^* + LE,$$  \hspace{1cm} (26a)

$$LM^* = LA_M F'.$$  \hspace{1cm} (26b)

This transformation is not needed in the LS-PCAF model due to the fact that we have two different component scores matrices, which can separately capture the implications of the different preprocessing procedures.

3.4. Goodness of fit

To evaluate the goodness of fit of the model we suggest the following index, that compares the explained sum of squares to the observed one by a percentage. Thus, we have

$$1 - \frac{(2^f + 1)\|M - M^*\|^2 + 2^f \|SA - S^*A\|^2}{(2^f + 1)\|M\|^2 + 2^f \|SA\|^2} \times 100,$$  \hspace{1cm} (27)

where $M^*$ is defined in (20b) and $S^*$ in (21) for CS-PCAF or in (23) for LS-PCAF. In the quotient term in (27), the numerator shows how much the estimated values of the centers and of the spreads differ from the original ones; the denominator gives the sum of squares of the centers and of the spreads according to the system of weights we used. The index takes values in the interval $[0, 100]$. 
4. Estimation procedure: an alternating least squares approach

In this section we propose an alternating least squares algorithm in order to solve the minimization problem of PCAF in (18). Indeed we propose two different algorithms to find the estimations of the parameters: one for CS-PCAF and one for LS-PCAF. We note that the following procedures do not prohibit the estimated spreads to become negative, even though this turns out to happen rarely. Indeed, we are working on a modified version of the algorithm that does guarantee non-negative spread estimates (e.g., compare in the framework of the fuzzy regression analysis, D’Urso, 2003). Currently, we suggest to replace negative estimate spreads by 0.

4.1. Estimation procedure for CS-PCAF

Upon substituting (22) and (20b) in (18) and taking into account the preprocessing procedure described in Section 3.3 we have

\[ L = R/SOH^2 xETX (A_M, F, B, D) \]

\[ = \| (2^J + 1)^{1/2}(LM - LAMF') \|^2 + \| (2^J)^{1/2}(S\Lambda M'BA - I_{J \times J} DA) \| \].

In the following, we give the updating of the matrices with respect to which we minimize expression (28). When we update with respect to a matrix, we solve the minimization problem in (28) keeping fixed the other matrices. To indicate the matrices that we currently use to minimize the function in (28) we will write only that matrix within the parentheses in \( L = R/A_M^2(\cdot) \). Whenever a matrix is updated, the loss function to minimize decreases. After updating all the matrices, if the value of the loss function decreased less than a specified percentage from the previous function value (in the simulation in the following section we use 0.0001%), we consider the algorithm converged, otherwise we repeat the updating of all the matrices involved. The expression in (28) has a lower bound and therefore the function value converges to a stable value.

Before updating, we have to initialize the matrices. The matrices B and D are initialized as diagonal matrices whose elements are randomly generated. The component scores matrix \( A_M \) and the component loadings matrix F are initialized using randomly generated values or performing the SVD on the matrix of the observed centers M.

**Updating \( A_M \)**: From (28), after defining \( Y_1 = S\Lambda A - I_{J \times J} DA \), it remains to minimize

\[ L = R/A_M^2(A_M) \]

\[ = \| (2^J + 1)^{1/2}(LM - LAMF') \|^2 + \| (2^J)^{1/2}(Y_1 - A_MF'BA) \|^2 \]

\[ = \| \begin{bmatrix} (2^J + 1)^{1/2}vec(LM) \\ (2^J)^{1/2}vec(Y_1) \end{bmatrix} - \begin{bmatrix} (2^J + 1)(F \otimes L) \\ (2^J)^{1/2}(ABF \otimes I_J) \end{bmatrix} vec(A_M) \|^2 \]

\[ = \| E_1 - E_2 vec(A_M) \|^2, \quad (29) \]
where $\text{vec}(X)$ is the operator that transforms the $m \times n$ matrix $X$ whose $i$th column is $x_i$ in an $mn$-dimensional vector $[x_i' \ldots x_n']'$.

\[
E_1 = \begin{bmatrix}
(2J + 1)^{1/2}\text{vec}(LM) \\
(2J)^{1/2}\text{vec}(Y_1)
\end{bmatrix}
\quad\text{and}\quad
E_2 = \begin{bmatrix}
(2J + 1)^{1/2}(F \otimes L) \\
(2J)^{1/2}(ABF \otimes I_j)
\end{bmatrix}.
\]

The minimization problem in (29) has the solution

\[
\text{vec}(A_M) = (E_2'E_2)^{-1}E_2'E_1.
\]

**Updating $F$:** From (28) we have

\[
L_R \tilde{A}_M^2(F) = \|(2J + 1)^{1/2}(LM - LA_MF')\|^2 + \|(2J)^{1/2}(Y_1 - A_MF'BA)\|^2
\]

\[
= \left\| \begin{array}{c}
(2J + 1)^{1/2}\text{vec}(LM) \\
(2J)^{1/2}\text{vec}(Y_1)
\end{array} \right\| - \left\| \begin{array}{c}
(2J + 1)^{1/2}(I_j \otimes LA_M) \\
(2J)^{1/2}(AB \otimes A_M)
\end{array} \right\| \text{vec}(F') \right\|^2
\]

\[
= \|E_1 - E_3\text{vec}(F')\|^2,
\]

where

\[
E_3 = \begin{bmatrix}
(2J + 1)^{1/2}(I_j \otimes LA_M) \\
(2J)^{1/2}(AB \otimes A_M)
\end{bmatrix}.
\]

The problem in (31) has the solution given by

\[
\text{vec}(F') = (E_3'E_3)^{-1}E_3'E_1.
\]

**Updating $B$ (diagonal):** In the first term of (28) the matrix $B$ is absent. Thus we can consider this term as a constant in the minimization problem and the term $2J$ is no longer important. Therefore, we consider

\[
L_R \tilde{A}_M^2(B) = \|(Y_1 - A_MF'BA)\|^2 = \|(Y_1 - A_MF'AB)\|^2 = \|(Y_1 - Y_2B)\|^2,
\]

where we use that the matrices $B$ and $A$ are diagonal and $Y_2 = A_MF'A$. The solution of the minimization problem in (33) is given by

\[
B = [\text{diag}(Y_1'Y_2)]^{-1}\text{diag}(Y_1'Y_2),
\]

where $\text{diag}(X)$ is the operator that transforms the square matrix $X$ of order $n$ in a diagonal matrix of the same dimension whose elements in the main diagonal are the same of $X$.

**Updating $D$ (diagonal):** The way to derive the update of $D$ is similar to that for $B$. Thus, from (28) we have

\[
L_R \tilde{A}_M^2(D) = \|S_A - A_MF'BA - I_{J\times J}DA\|^2 = \|Y_3 - Y_4D\|^2,
\]

where we use that the matrices $D$ and $A$ are diagonal and $Y_3 = S_A - A_MF'BA$ and $Y_4 = I_{J\times J}A$. The solution of the minimization problem in (35) is

\[
D = [\text{diag}(Y_4'Y_4)]^{-1}\text{diag}(Y_3'Y_4).
\]
To summarize, the estimation procedure has the following steps:

**Step 1:** Initialize the matrices $A_M$, $F$, $B$ and $D$ and calculate the loss function value.

**Step 2:** Update the matrices $A_M$, $F$, $B$ and $D$.

**Step 3:** Compute the new loss function value and check if it differs less than a specified value with respect to the previous loss function value: in this case the algorithm is considered converged; otherwise go to Step 2.

### 4.2. Estimation procedure for LS-PCAF

The estimation procedure for LS-PCAF is similar to the one for CS-PCAF but now we have to find the minimum of the loss function in (37) updating, alternatingly, the component scores matrices for the centers ($A_M$) and for the spreads ($A_S$), and the component loadings matrix $F$. To initialize the matrices $A_M$ and $F$, we propose to use randomly generated values or rational values performing the SVD on $M$. Similarly, with respect to the matrix $A_S$, we can use randomly generated values or rational values performing the SVD on the matrix of observed spreads $S$. Upon substituting (23) and (20b) in (18) we have

$$L = R/SOH^2 x ETX (A_M, A_S; F) = (2J + 1)\|M - A_M F'\|^2 + 2J\|SA - A_S F'A\|^2. \quad (37)$$

We can obtain updates of the component scores matrices $A_M$ and $A_S$ taking into account that the above matrices appear only in one of the two terms in (37).

**Updating $A_M$:** From (37) it remains to minimize

$$L = R A_M^2(A_M, A_S, F) = (2J + 1)\|M - A_M F'\|^2$$

and the solution is

$$A_M = MF(F'F)^{-1}. \quad (39)$$

**Updating $A_S$:** From (37) it remains to minimize

$$L = R A_S^2(A_S) = \|SA - A_S F'A\|^2. \quad (40)$$

In this case we obtain

$$A_S = SAAF(F'AAF)^{-1} = SA^2F(F'AAF)^{-1}. \quad (41)$$

**Updating $F$:** We have:

$$L = R A_F^2(F) = \| (2J + 1)^{1/2}(M - A_M F') \|^2 + \| (2J)^{1/2}(SA - A_S F'A) \|^2$$

$$= \left\| \begin{bmatrix} (2J + 1)^{1/2}vec(M) \\ (2J)^{1/2}vec(SA) \end{bmatrix} - \begin{bmatrix} (2J + 1)^{1/2}(I_J \otimes A_M) \\ (2J)^{1/2}(A \otimes A_S) \end{bmatrix} vec(F') \right\|^2$$

$$= \|E_4 - E_5 vec(F')\|^2 \quad (42)$$

where

$$E_4 = \begin{bmatrix} (2J + 1)^{1/2}vec(M) \\ (2J)^{1/2}vec(SA) \end{bmatrix} \quad \text{and} \quad E_5 = \begin{bmatrix} (2J + 1)^{1/2}(I_J \otimes A_M) \\ (2J)^{1/2}(A \otimes A_S) \end{bmatrix}.$$
The solution of the minimization problem in (42) is
\[ \text{vec}(F') = (E'_6 E_5)^{-1} E'_6 E_4. \] (43)

To summarize, the estimation procedure has the following steps:

\textbf{Step 1:} Initialise the matrices \( A_M, A_S \) and \( F \) and calculate the loss function value.

\textbf{Step 2:} Update the matrices \( A_M, A_S \) and \( F \).

\textbf{Step 3:} Compute the new loss function value and check if it differs less than a specified value with respect to the previous loss function value: in this case the algorithm is considered converged; otherwise go to Step 2.

4.2.1. The special case in which \( \Lambda \propto I \)

As we noted in Section 1, the triangular membership function in (4) is the most common one. In this case, using (5)–(7), the parameter \( \lambda \) is equal to 1/2. Therefore, in many situations, we can have a diagonal matrix \( \Lambda \) whose diagonal values are the same. Moreover, \( \Lambda \) is proportional to the identity matrix whenever the fuzzy variables have the same membership functions. Thus, we can often have
\[ \Lambda = aI_f, \] (44)
for \( a \in \mathbb{R} \) and \( a \neq 0 \). When the expression in (44) holds, the iterative algorithm in Section 4.2 is not necessary. By using (44), we can rewrite (37) as
\[ L = R_A^2(A_M, A_S, F) = (2^j + 1)\|M - A_M F'\|^2 + 2^j\|aS - aA_S F'\|^2 \]
\[ = \left\| \begin{pmatrix} (2^j + 1)^{1/2}M \\ (2^j)^{1/2}aS \end{pmatrix} - \begin{pmatrix} (2^j + 1)^{1/2}A_M \\ (2^j)^{1/2}aA_S \end{pmatrix} F' \right\| ^2 \]
\[ = \left\| E_6 - \begin{pmatrix} (2^j + 1)^{1/2}A_M \\ (2^j)^{1/2}aA_S \end{pmatrix} F' \right\| ^2, \] (45)
where
\[ E_6 = \begin{bmatrix} (2^j + 1)^{1/2}M \\ (2^j)^{1/2}aS \end{bmatrix}. \]

The estimates of \( A_M, A_S \) and \( F \) can be obtained by performing the SVD of \( E_6 \). It is well known—see, for instance, Ten Berge (1993)—that \( E_6 \) can be decomposed as
\[ E_6 = P_P Q_P R_P', \] (46)
where \( P_P = \begin{bmatrix} p_{1P}^j \\ p_{2P}^j \end{bmatrix} \) and \( R_P \) are matrices containing the first \( P \) unit length singular vectors of \( E_6 \) and \( Q_P \) is the diagonal matrix with on the diagonal the first \( P \) singular values of \( E_6 \). \( P_P \) is partitioned in two submatrices which are, respectively, \( P_P^1 \) and \( P_P^2 \), containing the first \( I \) rows and the last \( I \) rows of \( P_P \). From (46) we can obtain the estimate of the component loadings matrix \( F \) as
\[ F = R_P'. \] (47)
With respect to the component scores matrices, respectively for the centers and for the spreads, we have
\[
(2^J + 1)^{1/2} A_M = P_P^1 Q_P, \tag{48}
\]
\[
(2^J)^{1/2} a A_S = P_P^2 Q_P \tag{49}
\]
and, therefore, we obtain the following estimates
\[
A_M = (2^J + 1)^{-1/2} P_P^1 Q_P, \tag{50}
\]
\[
A_S = (2^J)^{-1/2} a^{-1} P_P^2 Q_P \tag{51}
\]

5. Plotting procedure to display the observation units

In PCAF we think that it is very useful and interesting to plot each unit represented in \( \mathcal{R}^J \) as a hypercube in the low dimensional space \( \mathcal{R}^P \). Especially if \( P = 2 \), we can represent each unit as a rectangle. One of the aims of PCAF is to offer a simpler graphical description of each unit. In this section we suggest two procedures to plot the observation units on the subspace spanned by the columns of \( \mathbf{F} \).

In the algorithm in the previous section we did not make any assumption about orthogonality constraints for the loadings matrix \( \mathbf{F} \) but if the matrix \( \mathbf{F} \) is not columnwise orthonormal the representation of each point—and, therefore, of each hypercube—is distorted (see, e.g., Kiers, 2000). However, by a general orthonormalization procedure (for instance Gram-Schmidt) we can find a transformation matrix \( \mathbf{T} \) for which \( \tilde{\mathbf{F}} = \mathbf{FT} \) is columnwise orthonormal. In this way we find an orthonormal basis that spans the subspace in which we represent each observation unit. The rotation of \( \mathbf{F} \) by using \( \mathbf{T} \) must be compensated by postmultiplying the component scores matrices \( A_M \) and \( A_S \) by \((\mathbf{T}')^{-1}\). These transformations do not modify the estimates of the centers and of the spreads, as can be seen as follows. Using the symbol \( \tilde{\text{tilde}} \) to indicate the matrices \( \tilde{\mathbf{F}}, A_M \) and \( A_S \) modified by \( \mathbf{T} \), the following relations hold:
\[
\tilde{A}_M \tilde{\mathbf{F}}' = A_M (\mathbf{T}')^{-1} \mathbf{T}' = A_M \mathbf{F}' = \mathbf{M}', \tag{52}
\]
\[
\tilde{A}_S \tilde{\mathbf{F}}' = A_S (\mathbf{T}')^{-1} \mathbf{T}' \mathbf{F}' = A_S \mathbf{F}' = \mathbf{S}', \tag{53}
\]
\[
\tilde{A}_M \tilde{\mathbf{F}}' \mathbf{B} + \mathbf{I}_{1 \times J} \mathbf{D} = A_M \mathbf{F}' \mathbf{B} + \mathbf{I}_{1 \times J} \mathbf{D} = \mathbf{S}', \tag{54}
\]
where (53) is used in the LS-PCAF model and (54) in the CS-PCAF model. It follows that, after applying the alternating least squares algorithm given in Section 4, we have to apply, for instance, the Gram-Schmidt procedure to obtain a columnwise orthonormal matrix \( \tilde{\mathbf{F}} \) which is the basis of the subspace on which we can project correctly each hypercube. We can now introduce the two plotting procedures.

In the first, we project the estimated vertices of each hypercube. The estimated vertices of the hypercube associated with the generic unit \( i \) are the elements of the matrix
\[
\mathbf{I}_{K \times J} M^*_i + H_{K \times J} S^*_i, \tag{55}
\]
where \( M_i^* \) and \( S_i^* \) are diagonal matrices of order \( J \) whose main diagonal elements are estimated centers and estimated spreads pertaining to the unit \( i \) and \( I_{K \times J} \) is a matrix of order \( K \times J \) with unit elements. Thus, the diagonal elements of \( M_i^* \) are the elements of the \( i \)th row of the matrix defined in (20b) and the ones of \( S_i^* \) are from the \( i \)th row of the matrix defined in (21), using the CS-PCAF model, or (23), using the LS-PCAF model. Finally \( H_{K \times J} \) is the matrix of order \( K \times J \), defined in Section 3. Thus, the coordinates of the vertices pertaining to the observation unit \( i \) on the subspace spanned by the columns of \( \tilde{F} \) are obtained as

\[
\tilde{A}_i = (I_{K \times J} M_i^* + H_{K \times J} S_i^*) \tilde{F},
\]

which is similar to the expression used in PCA, \( \tilde{A} = X^* \tilde{F} \), where \( \tilde{A} \) and \( \tilde{F} \) are, respectively, the component scores matrix and the columnwise orthonormal component loadings matrix and \( X^* \) is the estimated data matrix. By using expression (56), for \( i = 1, \ldots, I \), we can plot each observation unit as a hypercube in the low dimensional space spanned by \( \tilde{F} \).

Using the above procedure, the projected vertices do not define a rectangle (if we extract two components) or a hypercube (if we extract more than two components). This is a well-known problem recognized, for instance, by Bock and Diday (2000). In the same way as Bock and Diday (2000), we suggest to solve this problem by representing the generic unit \( i \) on each axis by the segment that includes all the projections. Thus, with our first plotting procedure we find hypercubes covering the projected hypercubes.

In addition, we propose a second way to represent the observation units. Unlike the above one, this procedure offers a representation of the observation units as exact hypercubes in the low dimensional space but has a drawback as we will point out later. To reach our purpose, we propose to use a matrix similar to \( H_{K \times J} \) whose order depends on the number of components \( P \). We refer to this matrix of order \( L \times P \) where \( L = 2^P \) as \( H_{L \times P} \). Each row of \( A_M \) gives the coordinates of the center of the hypercube associated with the \( i \)th observation unit on the subspace spanned by \( \tilde{F} \). In a similar way, the rows of \( A_S \) (in LS-PCAF) give the coordinates of the spreads of the \( I \) units; likewise for CS-PCAF we use \( \tilde{A}_S = S^* \tilde{F} \), that gives coordinates of \( S^* \) projected on \( \tilde{F} \).

Then, the coordinates of the vertices of the \( i \)th hypercube, for \( i = 1, \ldots, I \), in the low dimensional space spanned by \( \tilde{F} \) can be obtained as

\[
\tilde{A}_i = I_{L \times P}(\tilde{A}_m)_i + H_{L \times P}(\tilde{A}_s)_i,
\]

in the CS-PCAF model, and as

\[
\tilde{A}_i = I_{L \times P}(\tilde{A}_m)_i + H_{L \times P}(\tilde{A}_s)_i,
\]

in the LS-PCAF model. In (57) and (58), \( I_{L \times P} \) is the matrix of order \( L \times P \) with unit elements and \( (\tilde{A}_m)_i \), \( (\tilde{A}_s)_i \) and \( (\tilde{A}_s)_i \) are diagonal matrices of order \( P \) whose diagonal is the \( i \)th row of, respectively, \( \tilde{A}_M \), \( \tilde{A}_S \) and \( \tilde{A}_S \), for \( i = 1, \ldots, I \). The drawback in formulas (57) and (58) is the possibility that the coordinates in \( (\tilde{A}_s)_i \) and \( (\tilde{A}_s)_i \) can be negative. From a graphical point of view, the drawback vanishes because we always obtain a low dimensional hypercube: in this way it is as if we ignore the sign of the elements of \( (\tilde{A}_s)_i \) and \( (\tilde{A}_s)_i \). To avoid the drawback, we suggest to find, if possible,
a rotation matrix \( V \) so that \( \tilde{F} = \tilde{F}V \) is also columnwise orthonormal and \( \tilde{\tilde{A}}_S = \tilde{\tilde{A}}_S V \) in the LS-PCAF model (or \( \tilde{\tilde{A}}_S = \tilde{\tilde{A}}_S V \) in the CS-PCAF model) has positive elements.

6. Simulation study

In this section we give the results of a simulation study carried out to assess how PCAF works. In particular the simulation study aims to answer three questions:

1. Does PCAF recover the underlying structure in the data better than the classical PCA applied to the centers matrix?
2. Are the algorithms (for CS-PCAF and LS-PCAF) efficient?
3. Do the algorithms hit local optima frequently?

To answer the above questions, we have randomly generated fuzzy data sets with a known underlying factor structure and we have studied whether PCAF better recovers it than PCA.

After generating \( \tilde{\tilde{A}}_M \) and \( \tilde{F} \) randomly from the uniform distribution, respectively, the known component scores matrix and the component loadings matrix of the centers, we have constructed the data from these matrices and added noise. We have considered seven different levels of noise added (\( \alpha = 0.1, 0.3, 0.5, 0.75, 1.0, 1.5, 2.0 \)). In the same way, from the uniform distribution, we have constructed the spreads according to the two hypotheses of CS-PCAF and LS-PCAF with some noise added. Moreover, we have considered three different relative sizes of the spreads with respect to the one of the centers (\( \tau = 0.2, 0.5, 1.0 \)). For instance, \( \tau = 0.2 \) means that the size of the spreads is 0.2 times the one of the centers. Specifically, we have constructed centers and spreads according to

\[
M = \tilde{\tilde{A}}_M \tilde{F}' + \alpha N_M, \tag{59}
\]

\[
S = \tilde{\tilde{A}}_S \tilde{F}' + \alpha N_S, \tag{60}
\]

\[
S = (\tilde{\tilde{A}}_M \tilde{F}' \tilde{B} + \mathbf{1}_{I \times J} \tilde{D}) + \alpha N_S, \tag{61}
\]

where \( \tilde{\tilde{A}}_S \) and \( \tilde{\tilde{B}} \) and \( \tilde{\tilde{D}} \) are, respectively, the known component scores matrix for the spreads (in the LS-PCAF model), and the known diagonal matrices whose elements connect, linearly, centers and spreads (in the CS-PCAF model); matrices \( \tilde{\tilde{A}}_S \) and \( \tilde{\tilde{B}} \) and \( \tilde{\tilde{D}} \) were generated randomly from the uniform distribution. \( N_M \) and \( N_S \) were randomly generated matrices from the uniform distribution such that

\[
\| \tilde{\tilde{A}}_M \tilde{F}' \|^2 = \| N_M \|^2, \tag{62}
\]

\[
\| \tilde{\tilde{A}}_M \tilde{F}' \tilde{B} + \mathbf{1}_{I \times J} \tilde{D} \|^2 = \| N_S \|^2, \tag{63}
\]

\[
\| \tilde{\tilde{A}}_S \tilde{F}' \|^2 = \| N_S \|^2, \tag{64}
\]
where (62) is used in both models, (63) in CS-PCAF and (64) in LS-PCAF. The constraints in (62)–(64) allow us to quantify exactly the relative amount of noise using the parameter $\gamma / VT$.

We have considered four different analysis methods. The first two pertain to CS-PCAF where in the first one we have assumed that the matrix $x^T x$ had the same diagonal elements ($\lambda_j = 1/2$, for $j = 1, \ldots, J$) and, in the last one, the first half of the elements of $\Lambda$ was equal to $1/2$ (the fuzzy variables have a triangular membership function) and the last half to $\sqrt{\pi}/2$ (that is using a normal membership function). The same distinction holds for the last two methods in which we have considered the same structures for $\Lambda$ while using LS-PCAF.

We have constructed data sets of three different dimensions ($18 \times 8, 36 \times 8, 36 \times 16$). We have considered an underlying structure with two and three components ($P = 2, 3$). For each condition, we have constructed ten data sets. Therefore, the simulation was based on $3$ (relative sizes of the spreads) $\times$ $7$ (percentages of added noise) $\times$ $3$ (dimensions of the data set) $\times$ $2$ (components extracted) $\times$ $10$ (generated data sets for each condition) $\times$ $4$ (used models and hypotheses on the matrix $x^T x$) = 5040 randomly generated fuzzy data sets.

The data sets constructed according to (59) and (60) were analyzed by means of LS-PCAF, while those constructed according to (59) and (61) by means of CS-PCAF. Furthermore, the centers of all data sets have been analyzed by PCA as a reference.

For each case, the algorithm was run using one rational start, the SVD of the centers matrix $M$, and five random starts for LS-PCAF and one random start for CS-PCAF. We used only one random start in case of CS-PCAF to reduce the computation time of the simulation study that is relatively high for CS-PCAF. We worked in this way after noting that CS-PCAF very often attains the global optimum using the rational start, as was found in a smaller simulation performed on two generated data sets for each condition using one rational start and five random starts. In fact, in that study we found that in 97.4% of the cases the rationally started run led to the best solution.

To compare how the methods recover the underlying structure (question 1), we used two different indices based upon the differences between the known component loadings matrix and the estimated one. To deal with the rotational freedom in the model, in both cases, we first had to solve the following problem with respect to $T$.

$$f(T) = \min \| F^*_s T - \tilde{F} \|^2, \quad (65)$$

for $s = 1, 2$ where $\tilde{F}$ is the known component loadings matrix, as defined above, and $F^*_s$ is the estimated component loadings matrix which, for $s = 1$, is obtained by using one of the two PCAF models and, for $s = 2$, by using the classical PCA model. Finally, using the rotational freedom, $T$ is a transformation matrix of appropriate order, which transforms $F^*_1$ to $\tilde{F}^*_1 = F^*_1 T$ such that it becomes as similar as possible to $\tilde{F}$.

The solution of the problem in (65) is given by ordinary regression so, for $s = 1, 2$, $T = (F^*_s F^*_s)^{-1} F^*_s \tilde{F}$. It follows that we can find which one of the two methods recovers the underlying structure best by considering the following indices, for $s = 1, 2$:

$$g_1(s) = 1 - \frac{\| F^*_s - \tilde{F} \|^2}{\| \tilde{F} \|^2} \quad (66)$$
and

\[ g_2(s) = \frac{\sum_{j=1}^{J} \sum_{p=1}^{P} |\tilde{F}_{ip} - \tilde{f}_{ip}|}{JP}. \]  

(67)

In the index in (66), known as the proportion of recovery (PR) measure (e.g., Timmerman and Kiers, 2000), whose values are in [0, 1], we compare the squared norm of the difference between the estimated component loadings matrix \( \tilde{F} \) and the known one, and the squared norm of \( \tilde{F} \). In (67) we consider the mean absolute difference (MAD) between every element of \( \tilde{F} \) and the estimated one of \( \tilde{F}^* \). In both indices, for each case, we have used the values pertaining to the run of the algorithm that led to the smallest function value (in LS-PCAF in which \( \Lambda \) has equal diagonal elements we use the values pertaining to the run of the non-iterative algorithm).

We decided to conclude that one method performed noticeably better than the other when the difference on either of the indices exceeded a threshold value \( \psi \), usually \( \psi = 0.02 \). If the difference was smaller, we considered the difference to be negligible.

We have also considered the threshold value \( \psi = 0.10 \) in order to evaluate when one of the two methods recovers the underlying structure in the data in a much better way. In the sequel, when it is not specified, we refer to \( \psi = 0.02 \). To evaluate the efficiency of the algorithm, we compared the computation time both using the rational start and the random starts (question 2). It is useful to note that the simulation was carried out on a personal computer with Pentium III 1 GHz processor and 256 MB RAM. Finally, with respect to question 3, we studied the tendency of the algorithm to hit local optima. For this purpose, we checked, for each case, how many times the function value was more than 0.1% bigger than that of the optimal solution.

6.1. Results

6.1.1. Recovering performance of PCAF

In this section we give an answer to the first of the three questions proposed at the beginning of Section 6. The results are summarized in Table 1 that contains 36 rows and 10 columns. The first two columns refer to the condition involved, while, starting from the third, the other couples of columns refer to CS-PCAF with equal or different diagonal elements and to LS-PCAF with, again, equal or different diagonal elements. Each cell reports how many times one of the methods (PCAF in the first column, PCA in the second of each couple) worked noticeably better in terms of both indices. From Table 1 we can see that CS-PCAF and PCA performed similarly in very many cases. When \( \Lambda \) has equal diagonal elements, this was in particular the case if the level of noise added and the size of the spreads were jointly small, otherwise CS-PCAF worked slightly better than PCA. Considering CS-PCAF when \( \Lambda \) has different diagonal elements, the same results held, except that, if \( \tau = 1.0 \) and the level of noise added was high, it was found more often that PCA worked better than CS-PCAF than the other way around. Probably, in these conditions the relation between centers and spreads lost its validity. For LS-PCAF, the indices showed more remarkable differences and again, on the whole, we found that the performance of LS-PCAF was better than that of
Table 1
Results of CS-PCAF. Each cell shows the number of times in which one of the two methods works noticeably better than the other according to both indices. The values in the parentheses refer to the case $\psi = 0.10$ when they are not equal to 0.

<table>
<thead>
<tr>
<th>Condition</th>
<th>CS-PCAF (equal $\lambda$'s)</th>
<th>CS-PCAF (different $\lambda$'s)</th>
<th>LS-PCAF (equal $\lambda$'s)</th>
<th>LS-PCAF (different $\lambda$'s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PCAF</td>
<td>PCA</td>
<td>PCAF</td>
<td>PCA</td>
</tr>
<tr>
<td>$\tau = 0.2$</td>
<td>18 x 8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>36 x 8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>36 x 16</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\tau = 0.5$</td>
<td>18 x 8</td>
<td>3</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>36 x 8</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>36 x 16</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$\tau = 1.0$</td>
<td>18 x 8</td>
<td>11</td>
<td>8</td>
<td>16 (1)</td>
</tr>
<tr>
<td></td>
<td>36 x 8</td>
<td>12 (3)</td>
<td>4</td>
<td>15 (1)</td>
</tr>
<tr>
<td></td>
<td>36 x 16</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

PCA. In fact, we found that this only did not hold when the level of noise added was small (especially when $\Lambda$ has different diagonal elements). With respect to the other conditions (number of components extracted, data dimension, size of the spreads) we found few differences and, in general, LS-PCAF worked better than PCA.
Fig. 1. Average computation time of CS-PCA when \( \mathbf{A} \) has equal diagonal elements using the rational start and the random starts. The average computation time is distinguished with respect to the data size, the level of noise added, the number of extracted components and the size of the spreads.

In sum, to answer the first question, we can state that LS-PCA recovers the underlying structure in the data better than PCA. With respect to CS-PCA, we can conclude that CS-PCA recovers the underlying structure in the data sometimes better and often equally well as PCA, except in rare conditions in which PCA is better. PCA works better if \( \mathbf{A} \) has different diagonal elements, high size of the spreads and high level of noise added. The last consideration (and the ones about LS-PCA) helps us to stress the role of the structure of \( \mathbf{A} \): it does not affect a lot the results but PCAF works better when \( \mathbf{A} \) has equal diagonal elements.

### 6.1.2. Efficiency of the algorithms

In this section we answer the second question in Section 6, pertaining to the efficiency of the algorithm. In Fig. 1, we give the average computation time of the algorithm, distinguishing the use of the rational start and of the random starts. For each case, the algorithm was run using the rational start and the random starts. In the following figures, the average computation time for each value of the parameters is reported. The average computation time pertaining to the random starts is the mean of the computation times of all five runs (one run for CS-PCA) for all ten cases for each condition.

With respect to LS-PCA, in which \( \mathbf{A} \) has equal diagonal elements, we run the non-iterative algorithm of Section 4.2.1. Therefore, in this section, we did not consider the case of LS-PCA, in which \( \mathbf{A} \) has equal diagonal elements.

Using a rational start (results displayed in Fig. 1) the average computation time for the procedure based on CS-PCA with \( \mathbf{A} \) having equal diagonal elements was smaller
Fig. 2. Average computation time of CS-PCAF when Λ has different diagonal elements using the rational start and the random starts. The average computation time is distinguished with respect to the data size, the level of noise added, the number of extracted components and the size of the spreads.

(15.3 s) than that using random starts (60.6 s). Considering both kinds of start, the computation time increased if the data dimension or the number of extracted components or the size of the spreads increased. More or less the same comments hold for CS-PCAF with Λ having different diagonal elements, as we can observe considering Fig. 2. In this case, the differences between the rationally started run and randomly started runs were smaller than above: a bit more than doubled was the average computation time using the rational start (42.55) compared with that using random starts (140.15).

The average computation time was dramatically smaller for LS-PCAF for Λ having different diagonal elements (Fig. 3). In this case, the average computation time was 0.64 s using the rational start and 1.28 s using random starts. Again the average computation time increased if the dimension of the data or the number of extracted components or the size of the spreads increased. In this case, the level of noise and, above all, the number of variables implied an increase of the computation time too.

Now we can answer the question about the efficiency. The computation time of LS-PCAF is considerably smaller than that of CS-PCAF. The structure of the matrix Λ affects the computation time by increasing it if the diagonal elements are different. The runs using the rational start are more efficient than those using random starts. Finally, the efficiency of the algorithm decreases if the dimension of the data or the level of noise added or the number of extracted components or the size of the spreads increase, although for CS-PCAF the level of noise does not seem to affect the computation time much.
6.1.3. Frequency of hitting local optima of the algorithms

The last question in the beginning of Section 6 was ‘Do the algorithms hit local optima frequently?’. As in the previous subsection we answer it by analyzing the results of each analysis method distinguishing between the rational start and random starts.

The results are reported in Figs. 4–7. The points in Figs. 4 and 6 display how many times on average, in each condition, the runs of the algorithm using the rational start lead to the global optimum (i.e., if the function value is less than 0.1% bigger than the lowest one obtained in the six runs). Fig. 4 refers to CS-PCAF and the average values can range from 0 to 2, because the results are based on a smaller simulation study performed on two data sets for each condition. Instead, Fig. 6 refers to LS-PCAF and the values can range from 0 to 10. Finally, in Figs. 5 and 7, respectively, for CS-PCAF and LS-PCAF, the points report the average number of random starts that led to the global optimum in each condition (the value can range from 0 to 5).

Again, we considered LS-PCAF only when \( \Lambda \) has different diagonal elements. The frequency of hitting local optima depended, above all, on the algorithm run but also on the size of the spreads, the structure of \( \Lambda \) and the level of noise. The dimension of the data matrix and the number of extracted components were not important.

We first consider CS-PCAF. We almost always obtained the global optimum using the rational or the random starts, both when \( \Lambda \) has equal diagonal elements and when \( \Lambda \) has different diagonal elements. In fact, the results were often very good except in the condition \( \tau = 1.0 \) where the frequency of hitting local optima was still good but the worst of the entire simulation, especially when \( \Lambda \) has different diagonal elements.

The results of the algorithm for LS-PCAF were even better than the previous ones. With respect to the rational start, the algorithm led to a local optimum a
negligible number of times. In the other case—algorithm runs using the random starts—the frequency of hitting global optima was still very high.

To conclude, we can answer the last question observing that, in the algorithms for LS-PCAF and CS-PCAF, we can use only the rational start (remembering that the
average computation time is lower) because it leads practically always to the global optimum (the risk is lower than 1%). The one condition in which more than the rational start could be advisable is if the size of the spreads is high and $A$ has different diagonal elements, because then the probability to encounter local optima often is considerably higher (about 5%).
7. Application to a real data set

In this section we apply PCAF to a real data set, introduced by Ichino (1988). The involved data set is known as ‘Fats and Oils data’ and is reproduced in Table 2.

The data set refers to eight oils ($I = 8$) described by four quantitative interval valued variables ($J = 4$). Thus, each oil is described by the ‘Specific Gravity’, the ‘Freezing Point’, the ‘Iodine Value’ and the ‘Saponification’. In fact, there is also a qualitative variable but we only refer to the quantitative ones.

We have assumed that all the variables have a triangular membership function and that a linear relation between centers and spreads cannot be a plausible assumption. Therefore, we have applied LS-PCAF. As $\mathbf{X}$ has equal diagonal elements, the solutions are obtained using the non-iterative algorithm proposed in Section 4.2.1.

Before running the algorithm, we have preprocessed the data in the way described in Section 3.3. We have extracted two components ($P = 2$) and the goodness of fit, using the index in (27), was 90.2%. After applying the Gram-Schmidt procedure to the matrix $\mathbf{F}$, we have obtained the component loadings matrix in Table 3.

Now we can adequately represent the eight oils to better understand their features. We have applied both plotting procedures given in Section 5. Using (56) and after finding the segment that includes all the vertices, we have obtained the representation given in Fig. 8.

Table 2
Fats and oils data set

<table>
<thead>
<tr>
<th>Fats and oils</th>
<th>Specific gravity</th>
<th>Freezing point</th>
<th>Iodine value</th>
<th>Saponification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linseed oil (LIN)</td>
<td>[0.930, 0.935]</td>
<td>[−27, −18]</td>
<td>[170, 204]</td>
<td>[118, 196]</td>
</tr>
<tr>
<td>Perilla oil (PER)</td>
<td>[0.930, 0.937]</td>
<td>[−5, −4]</td>
<td>[192, 208]</td>
<td>[188, 197]</td>
</tr>
<tr>
<td>Cottonseed oil (COT)</td>
<td>[0.916, 0.918]</td>
<td>[−6, −1]</td>
<td>[99, 113]</td>
<td>[189, 198]</td>
</tr>
<tr>
<td>Sesame oil (SES)</td>
<td>[0.920, 0.926]</td>
<td>[−6, −4]</td>
<td>[104, 116]</td>
<td>[187, 193]</td>
</tr>
<tr>
<td>Camellia oil (CAM)</td>
<td>[0.916, 0.917]</td>
<td>[−21, −15]</td>
<td>[80, 82]</td>
<td>[189, 193]</td>
</tr>
<tr>
<td>Olive oil (OLI)</td>
<td>[0.914, 0.919]</td>
<td>[0, 6]</td>
<td>[79, 90]</td>
<td>[187, 196]</td>
</tr>
<tr>
<td>Beef tallow (BEE)</td>
<td>[0.860, 0.870]</td>
<td>[30, 38]</td>
<td>[40, 48]</td>
<td>[190, 199]</td>
</tr>
<tr>
<td>Hog fat (HOG)</td>
<td>[0.858, 0.864]</td>
<td>[22, 32]</td>
<td>[53, 77]</td>
<td>[190, 202]</td>
</tr>
</tbody>
</table>

Table 3
Component loadings matrix after Gram-Schmidt procedure

<table>
<thead>
<tr>
<th>Variables</th>
<th>First component</th>
<th>Second component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specific gravity</td>
<td>−0.51</td>
<td>0.42</td>
</tr>
<tr>
<td>Freezing point</td>
<td>0.52</td>
<td>−0.19</td>
</tr>
<tr>
<td>Iodine value</td>
<td>−0.49</td>
<td>0.19</td>
</tr>
<tr>
<td>Saponification</td>
<td>0.47</td>
<td>0.86</td>
</tr>
</tbody>
</table>
We have plotted (in Fig. 9) the eight oils also by means of the second plotting procedure using formula (58). The only difference is that the rectangles obtained using (56) are slightly oversized especially with respect to the first component. Anyway we can note three clusters of oils. The first is composed by Linseed and Perilla (oils used for paint) and the second, the biggest one, by Cottonseed, Sesame, Camelia and Olive. Finally, very far from the other two clusters, the last one appears on the right. It is composed by Beef tallow and Hog fat, two animal fats.

Inspecting Figs. 8 and 9, we can observe that the obtained results are quite similar. According to the locations of the oils in the two figures and to the features of the oils we can interpret the axes. On the right (left) of the first axes there are oils whose Saponification and Freezing Point values are higher (lower). We can verify this by observing that the two fats on the right have the highest values (using the mid-point of the interval): respectively, 194.5 and 34 for the beef tallow and 196 and 27 for the Hog fat. The second axes is slightly more complex. On the top we can find the oils whose Specific Gravity value and, again, Saponification value are higher. For this reason, Perilla is on the top: it has the highest Specific Gravity value and the Saponification value is on average (it is the forth highest value). Linseed has a high Specific Gravity but a low Saponification and, on the contrary, Beef tallow and Hog fat have high Saponification values but very low Specific Gravity values (their location is the lowest for this is reason). The values of Saponification and Specific Gravity of the other four oils, pertaining to the second cluster, are on average. Thus, it explains their position in the middle. The important role played by Saponification in both axes helps...
also to explain the size of the rectangle pertaining to Linseed, which is by far the biggest. The fuzziness of this score is the biggest one: the lower bound is 118 and the upper bound 196 which lead to a center value equal to 157 and a spread value equal to 39.

8. Conclusion

In this paper we have proposed two PCA procedures to detect the underlying structure of fuzzy data of $I$ observation units and $J$ fuzzy variables. In the first (CS-PCAF), we assume that estimated spreads are linearly related to the estimated centers matrix. In the second (LS-PCAF) we assume that the spreads are decomposed in a component scores matrix (different from the one of the centers) and the same component loadings matrix of the centers. In this way, LS-PCAF searches a compromise structure between centers and spreads. Also, we suggested two procedures for plotting component scores. In the first, taking Bock and Diday (2000) as a starting point, we found the hypercubes (on the low dimensional space) including all the projected vertices of the original hypercubes. In the second, we have obtained exact hypercubes, but the procedure has a drawback and does not always have a solution. In this context, further research is necessary.

To assess how PCAF works, we have compared it with classical PCA by means of the results of an application to real fuzzy data and of a simulation in which we
randomly generated fuzzy data sets whose underlying structure was known. The data sets were constructed according to different levels of noise, data dimensions, extracted components, sizes of the spreads and membership functions. On the whole, the results have showed that the performance of LS-PCAF was better than that of PCA, but the performance varied across the above conditions. With respect to CS-PCAF, we found that it worked better than or equally well as PCA.

On the basis of the good results of the simulation and of the real application, we indicate at least two perspectives of research: improvements in the graphical representation of the units on the low dimensional space, as noted above, and the extension of the model to deal with three-way fuzzy data sets. Furthermore, it will be interesting to carry out a new simulation study in order to compare our methods to the work by Cazes et al. (1997) and extensions of their approach (Palumbo and Lauro, 2002) and to the work by Yabuuchi et al. (1997).

Acknowledgements

The authors are grateful to the Co-Editor and the Referees for their suggestions and comments.

References
