A COMPARISON OF THREE METHODS FOR PRINCIPAL COMPONENT ANALYSIS OF FUZZY INTERVAL DATA

PAOLO GIORDANI
DEPARTMENT OF STATISTICS, PROBABILITY AND APPLIED STATISTICS
UNIVERSITY OF ROME "LA SAPIENZA"
P.LE ALDO MORO 5, 00185 ROME, ITALY
PAOLO.GIORDANI@UNIROMA1.IT

HENK A.L. KIERS
HEYMANS INSTITUTE (DPMG)
UNIVERSITY OF GRONINGEN
GROTE KRUISSTRAAT 2/1, 9712 TS GRONINGEN, THE NETHERLANDS
H.A.L.KIERS@PPSW.RUG.NL

ABSTRACT

Vertices Principal Component Analysis (V-PCA), and Centers Principal Component Analysis (C-PCA) generalize Principal Component Analysis (PCA) in order to summarize interval valued data. Neural Network Principal Component Analysis (NN-PCA) represents an extension of PCA for fuzzy interval data. However, also the first two methods can be used for analyzing fuzzy interval data, but they then ignore the spread information. In the literature, the V-PCA method is usually considered computationally cumbersome because it requires the transformation of the interval valued data matrix into a single valued data matrix the number of rows of which depends exponentially on the number of variables and linearly on the number of observation units. However, Cazes et al (1997) have shown that this problem can be overcome by considering the cross-products matrix which is easy to compute. The present paper offers a review of C-PCA and V-PCA (which hence also includes the computational short-cut to V-PCA) and NN-PCA. Furthermore, a comparison is given of the three methods by means of a simulation study and by application to an empirical data set. In the simulation study, fuzzy interval data are generated according to various models, and it is reported in which conditions each method performs best.

KEYWORDS: Data reduction, Fuzzy Interval data, Principal Component Analysis.

1. Introduction

The need for fuzzy set theory emerged from the attempt of providing a rigorous mathematical framework for precisely dealing with uncertain phenomena. Here, the concept of uncertainty refers to situations characterized by absence of sharply defined criteria of class membership, as firstly noted by Zadeh (1965). Instead, interval analysis emerged from the desire to manage the inaccuracies of measuring instruments. Thus, fuzzy set theory and interval analysis arose from different needs, but the interrelationship between them is recognized. See, for more details, Lodwick and Jamison (2003).

A fuzzy set consists of fuzzy scores or measurement of objects on variables. These fuzzy scores are given in the form of "membership functions". Specifically, each score is defined fuzzily, as a function on a domain of possible values, where this function indicates for each possible value the degree of applicability (or certainty or belief) of this value to the object being measured. If the domain of possible values is X, then a membership function is a subset of $\Re^+$ such that $\sup_{x\in X} \mu_{\tilde{x}}(x) < \infty$, but, usually, $\mu_{\tilde{x}}(x) \in [0,1], \forall x \in X$ and $\sup_{x\in X} \mu_{\tilde{x}}(x) = 1$ (normal fuzzy sets). Currently, a fuzzy set is represented solely by its membership function. See, for more details, Zimmermann (2001).

A general class of fuzzy sets, that we will consider in this paper, is the so-called LR family. A fuzzy interval $\tilde{x} = (x_L^-, x^-, x^+, x_R^+)$, with $x_L^- < x^- < x^+ < x_R^+$, is of LR type if there exist two functions L (for left) and R (for right), which must fulfill particular requirements (see Zimmermann, 2001) with membership function

$$\mu_{\tilde{x}}(x) = \begin{cases} L\left(\dfrac{x^- - x}{x^- - x_L^-}\right) & x \le x^-, \\ 1 & x^- \le x \le x^+, \\ R\left(\dfrac{x - x^+}{x_R^+ - x^+}\right) & x \ge x^+. \end{cases} \tag{1}$$

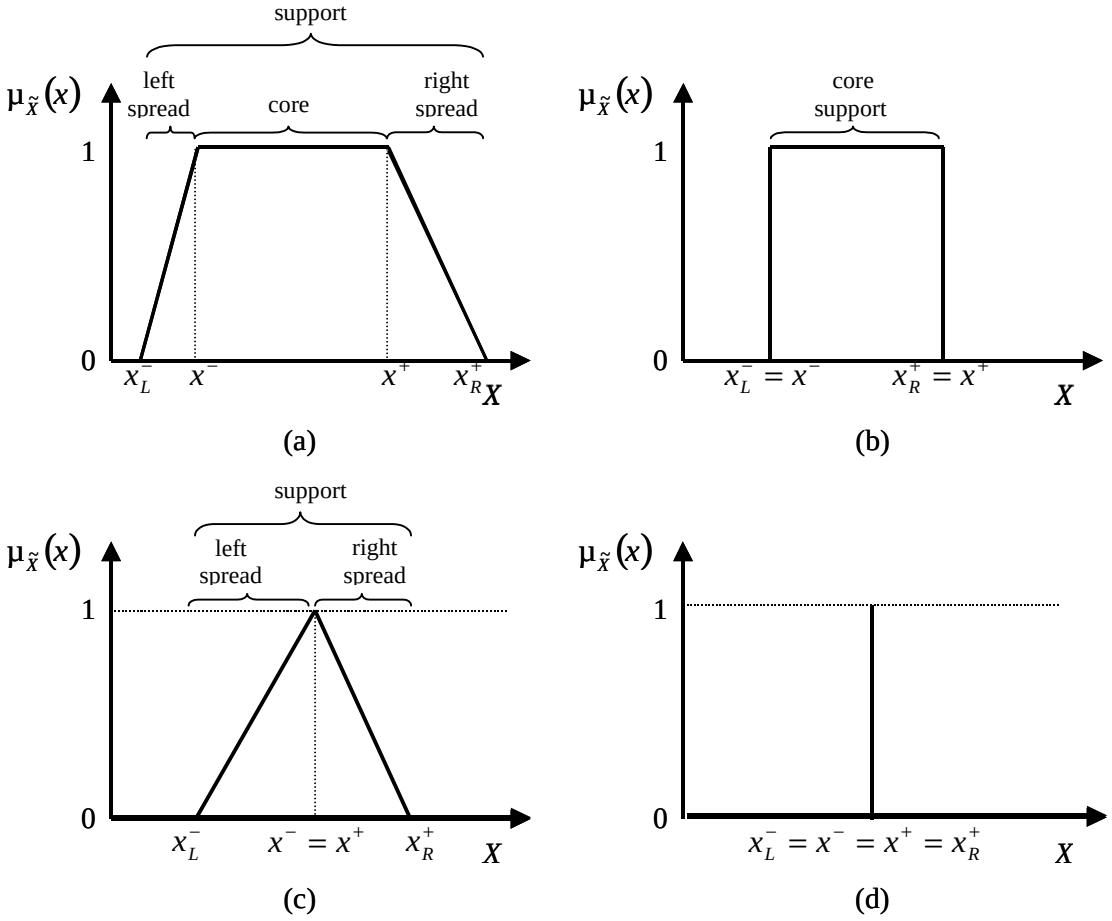
The quantities $(x^- - x_L^-)$ and $(x_R^+ - x^+)$ are the left and right spreads, respectively. The intervals $[x_L^-, x_R^+]$ and $[x^-, x^+]$ are usually denoted as the support and the core of a fuzzy interval, respectively. A classical non-fuzzy, or crisp, interval can be derived from a fuzzy interval when $x_L^- = x^-$ and $x^+ = x_R^+$ (when the spreads are equal to 0). In this case, (1) is replaced by

$$\mu_{\tilde{x}}(x) = \begin{cases} 1 & x^- \le x \le x^+, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

From (2), it is easy to see that differently from fuzzy sets, intervals have an 'all-or-nothing nature'. Interval analysis has no gradations: one does not know the exact value of the measurement, but one only knows the exact limit of its domain of variation. A number x is either in an interval X, or it is not and, thus, the concept of degree of membership vanishes. Therefore, we can state that interval analysis can be considered as a subdomain of fuzzy set theory. For the sake of completeness, it is

fruitful to observe that a fuzzy number and a crisp number are special cases of fuzzy interval when $x^- = x^+$ and $x_L^- = x^- = x^+ = x_R^+$, respectively. See also Figure 1.

Figure 1: *Membership functions of fuzzy intervals (a), crisp intervals (b), fuzzy numbers (c), crisp numbers (d).*



As with crisp data, it frequently occurs that one aims at summarizing the data through a low number of (unobservable) variables. Principal Component Analysis (PCA) is a well-known technique for summarizing crisp data. Specifically, let n and m be the numbers of, respectively, observation units and variables. PCA finds $p \leq m$ unobserved variables, called components, which are linear combinations of the observed ones, such that they capture the (observed) information as much as possible. The observed data matrix $\mathbf{X}$ is approximated by

$$\hat{\mathbf{X}} = \mathbf{A}\mathbf{B}', \tag{1}$$

where $\mathbf{A}$ ($n \times p$) is the component scores matrix and $\mathbf{B}$ ($m \times p$) the component loadings matrix. Note that $\hat{\mathbf{X}}$ provides the best p-rank approximation of $\mathbf{X}$. Also note that $\mathbf{A} = \mathbf{X}\mathbf{B}$ if the component loadings matrix is columnwise orthonormal.

Several authors generalized PCA to deal with interval valued and fuzzy data: concerning interval valued data, we refer to Centers Principal Component Analysis (C-PCA) and Vertices Principal Component Analysis (V-PCA) by Cazes et al. (1997), also see Bock and Diday (2000). Further works can be found in Lauro and Palumbo (2000), Cazes (2002), Palumbo and Lauro (2003) and D'Urso and Giordani

(2004). Concerning fuzzy data, as far as is known by the authors, at least five works are available in the literature. Watada, and Yabuuchi (1997), Giordani and Kiers (2004a) and D'Urso and Giordani (2005) extend PCA so as to handle fuzzy numbers. Instead Coppi et al. (2004) and Neural Networks Principal Component Analysis (NN-PCA) by Denœux and Masson (2004) attempt to generalize PCA so as to handle fuzzy interval data. It should be noted that methods for interval valued data can also be used to analyze fuzzy interval data by simply ignoring the spreads information. In other words, these methods then only use the cores of the fuzzy intervals. Alternatively, these methods could be applied to the supports, or to intervals midway between the cores and the supports. Vice versa, methods for fuzzy data can also be used for analyzing interval data, because the latter can be seen as a special case of fuzzy data, with spreads equal to 0.

In this paper, we shall compare C-PCA, V-PCA and NN-PCA. At the time of writing, the latter is the only one published work for PCA of fuzzy intervals. C-PCA and V-PCA are probably the two most popular methods for PCA of interval valued data and hence are chosen.Note also that C-PCA, V-PCA and NN-PCA have largely the same aim in summarizing the data: finding crisp loadings, and fuzzy (or interval valued) component scores of the ranges of data values.

The C-PCA method basically consists of a PCA on the centers of the intervals, whereas the V-PCA method is based on using all vertices of the hyperrectangle defined by the intervals for all variables for each observation unit (as explained below). V-PCA consists of a PCA on the ensuing matrix. The number of rows of this matrix depends exponentially on the number of variables and linearly on the number of observation units, and appears to make the actual computation of a PCA solution practically impossible even for relatively small numbers of variables. Indeed, Cazes et al (1997) have shown that this problem can be overcome by considering the cross-products matrix which is easy to compute. Nonetheless, V-PCA seems only to be commonly thought of as a method requiring the PCA of a huge matrix (see, e.g., Denœux and Masson, 2004, p. 337), which suggests to avoid performing V-PCA and rather use C-PCA instead. However, as Denœux and Masson note, C-PCA unfortunately "does not take into account the imprecision of the data in the feature extraction process and, consequently, builds only suboptimal low-dimensional representation of the data [...]". Finally, the NN-PCA method detects the underlying structure of fuzzy intervals by extending in a fuzzy framework the connection between standard PCA and autoassociative multilayer perceptrons described by Boulard and Kamp (1988).

C-PCA, V-PCA and NN-PCA have been described in the literature, but how to choose between these methods has received little or no attention. In the present paper, we offer empirical information for facilitating this choice, by comparing the methods by means of a simulation study and by application to an empirical data set. In the simulation study, fuzzy data are generated according to various models, and it is studied in which conditions each method performs best.

The paper is organized as follows. In Section 2 and in Section 3, we present V-PCA and C-PCA, respectively. In particular, in Section 2, we explain in detail the computational short-cut for the solution of the V-PCA method. In Section 4, the NN-PCA method is reviewed. Finally, Section 5 and Section 6 are devoted to, respectively, a simulation study carried out in order to compare the above three methods, and an application of all the methods to a fuzzy data set.

2. V-PCA: Vertices Principal Component Analysis

The Vertices Principal Component Analysis (V-PCA), proposed by Cazes et al. (1997), offers the possibility to detect the underlying structure of the two-way interval valued data set stored in $\mathbf{X}$ ($n \times m$):

$$\mathbf{X} = \begin{pmatrix} \left[x_{11}^-, x_{11}^+\right] & \cdots & \left[x_{1m}^-, x_{1m}^+\right] \\ \vdots & \ddots & \vdots \\ \left[x_{n1}^-, x_{n1}^+\right] & \cdots & \left[x_{nm}^-, x_{nm}^+\right] \end{pmatrix}. \tag{2}$$

The i-th row of $\mathbf{X}$ pertains to the i-th observation unit, $i = 1, \ldots, n$. As each observation unit is characterized by m (interval valued) variables, it can be represented as a hyperrectangle in $\Re^m$ and the number of vertices of each hyperrectangle is 2^m. If $m = 1$, each hyperrectangle is reduced to a segment and, if $m = 2$, to a rectangle.

V-PCA does not directly summarize the interval valued data in (2). In fact, (2) is replaced by a single valued data matrix obtained as follows. Each interval valued row is transformed into the numerical matrix $\mathbf{X}_i$:

$$\mathbf{X}_i = \begin{pmatrix} x_{i1}^- & x_{i2}^- & \cdots & x_{im}^- \\ x_{i1}^+ & x_{i2}^- & \cdots & x_{im}^- \\ \vdots & \vdots & \ddots & \vdots \\ x_{i1}^+ & x_{i2}^+ & \cdots & x_{im}^+ \end{pmatrix}. \tag{3}$$

$\mathbf{X}_i$ is obtained in such a way that each row refers to each vertex of the i-th hyperrectangle. If $m = 3$, we have

$$\mathbf{X}_i = \begin{pmatrix} x_{i1}^- & x_{i2}^- & x_{i3}^- \\ x_{i1}^+ & x_{i2}^- & x_{i3}^- \\ x_{i1}^- & x_{i2}^+ & x_{i3}^- \\ x_{i1}^- & x_{i2}^- & x_{i3}^+ \\ x_{i1}^- & x_{i2}^+ & x_{i3}^+ \\ x_{i1}^+ & x_{i2}^- & x_{i3}^+ \\ x_{i1}^+ & x_{i2}^+ & x_{i3}^- \\ x_{i1}^+ & x_{i2}^+ & x_{i3}^+ \end{pmatrix}. \tag{4}$$

Thus, $\mathbf{X}_i$, $i = 1, \ldots, n$, has 2^m rows and m columns. By stacking below each other all the matrices $\mathbf{X}_i$'s, $i = 1, \ldots, n$, we get the new numerical valued data matrix $\mathbf{X}_{V-PCA}$ with $n2^m$ rows and m columns:

$$\mathbf{X}_{V-PCA} = \begin{pmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_n \end{pmatrix}. \tag{5}$$

V-PCA consists of performing PCA on (5). As for ordinary PCA, it can advisable to preprocess the data in order to avoid unwanted differences among the variables. The matrix in (5) can be preprocessed as in the standard single valued case.

The application of PCA to the matrix $\mathbf{X}_{V-PCA}$ will give $\hat{\mathbf{X}}_{V-PCA} = \mathbf{A}_{V-PCA}\mathbf{B}'$, and if $\mathbf{B}$ is chosen to be columnwise orthonormal, we have $\mathbf{A}_{V-PCA} = \mathbf{X}_{V-PCA}\mathbf{B}$. To facilitate the interpretation of the solution, it is often useful to plot the observation units. When the loadings furnish an orthonormal basis, the component scores directly give the coordinates for a plot of the observation units projected on the space spanned by the components (e.g., see Kiers, 2000). In case of V-PCA, for each observation unit, we could analogously plot the component scores for all vertices for each observation unit, which would thus represent the projected hyperrectangles for all observation units. However, in this way, each projected hyperrectangle would be represented by a (large) number of projected vertices, and would not give a clear geometrical shape. A more attractive way of plotting would be to plot the rectangle (in two-dimensions), box (in three dimensions), or hyperrectangle in more than three dimensions, that envelops all projected vertices of a projected hyperrectangle. For this purpose, Cazes et al. (1997) proposed to determine, for each observation unit, for each component, the segment containing all component scores for vertices associated with this observation unit. Specifically, with respect to the *k*-th component, $k = 1,\ldots,p$, if n_i denotes the set of all the vertices for the *i*-th observation unit, $i = 1,\ldots,n$, the lower and upper bounds of such a segment are, respectively,

$$a_{ik}^{-} = \min_{l \in n_i}(a_{lk}), \tag{6}$$

$$a_{ik}^{+} = \max_{l \in n_i}(a_{lk}). \tag{7}$$

It should be clear that V-PCA implicitly takes into account the interval widths by considering all the vertices pertaining to each hyperrectangle. Unfortunately, it requires the analysis of a data matrix $\mathbf{X}_{V-PCA}$, see (5), the dimension of which is often huge. Specifically, as the number of columns increases, the number of rows increases exponentially. For example, suppose we deal with $n = 16$ observation units and $m = 12$ interval valued variables, then, for this relatively small data set $\mathbf{X}_{V-PCA}$ becomes huge because it has order 65536×12, which is hard to handle. Larger numbers of variables soon make the PCA of such a matrix $\mathbf{X}_{V-PCA}$ practically impossible.

Cazes et al. (1997) have shown that this computational problem can be overcome by considering a special property of PCA. Specifically, it is well-known that the columns of the component loadings matrix are the eigenvectors obtained from the eigendecomposition of the cross-products matrix. Note that the eigenvectors are arranged in such a way that the first ones are associated with the highest eigenvalues. Dealing with the cross-products matrix $\mathbf{C}_{V-PCA} = \mathbf{X}'_{V-PCA}\mathbf{X}_{V-PCA}$ has two relevant advantages. The (square and symmetric) matrix $\mathbf{C}_{V-PCA}$ has order *m*, that is the

number of observed variables. Moreover, $\mathbf{C}_{V-PCA}$ is very simple to compute without involving the computation of $\mathbf{X}_{V-PCA}$. In fact, let us consider two different columns of $\mathbf{X}_{V-PCA}$ (without loss of generality the first two columns) whose cross-products are stored in the off-diagonal elements of $\mathbf{C}_{V-PCA}$. These two columns have only two values: the first x_{i1}^- and x_{i1}^+, $i=1,\ldots,n$, and the second x_{i2}^- and x_{i2}^+, $i=1,\ldots,n$. The two columns are such that the four combinations (x_{i1}^-, x_{i2}^-), (x_{i1}^+, x_{i2}^-), (x_{i1}^-, x_{i2}^+), (x_{i1}^+, x_{i2}^+) occur equally often. More specifically, if there are two variables, each combination occurs only once, if there are three variables, twice, if there are four variables, four times, if there are *m* variables, each combination occurs 2^{m-2} times. This implies that, the cross-product of the first two columns is

$$2^{m-2}\sum_{i=1}^{n}\left(x_{i1}^-x_{i2}^- + x_{i1}^+x_{i2}^- + x_{i1}^-x_{i2}^+ + x_{i1}^+x_{i2}^+\right) = 2^{m-2}\sum_{i=1}^{n}\left(x_{i1}^- + x_{i1}^+\right)\left(x_{i2}^- + x_{i1}^+\right). \tag{8}$$

Let us consider the diagonal elements of $\mathbf{C}_{V-PCA}$. The first element refers to the first column of $\mathbf{X}_{V-PCA}$. The cross-product is the sum of the squared (lower and upper) bounds times a constant. In the general case of *m* variables, it is easy to see that this constant equals 2^{m-1}. Hence, we have

$$2^{m-1}\sum_{i=1}^{n}\left(x_{i1}^{-2} + x_{i1}^{+2}\right); \tag{9}$$

analogous expressions hold for the other columns of $\mathbf{X}_{V-PCA}$. Now taking into account (8) and (9), we get

$$\mathbf{C}_{V-PCA} = 2^{m-2}\begin{bmatrix} 2\sum_{i=1}^{n}\left(x_{i1}^{-2} + x_{i1}^{+2}\right) & \sum_{i=1}^{n}\left(x_{i1}^- + x_{i1}^+\right)\left(x_{i2}^- + x_{i2}^+\right) & \cdots & \sum_{i=1}^{n}\left(x_{i1}^- + x_{i1}^+\right)\left(x_{im}^- + x_{im}^+\right) \\ \sum_{i=1}^{n}\left(x_{i2}^- + x_{i2}^+\right)\left(x_{i1}^- + x_{i1}^+\right) & 2\sum_{i=1}^{n}\left(x_{i2}^{-2} + x_{i2}^{+2}\right) & \cdots & \sum_{i=1}^{n}\left(x_{i2}^- + x_{i2}^+\right)\left(x_{im}^- + x_{im}^+\right) \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n}\left(x_{im}^- + x_{im}^+\right)\left(x_{i1}^- + x_{i1}^+\right) & \sum_{i=1}^{n}\left(x_{im}^- + x_{im}^+\right)\left(x_{i2}^- + x_{i2}^+\right) & \cdots & 2\sum_{i=1}^{n}\left(x_{im}^{-2} + x_{im}^{+2}\right) \end{bmatrix}. \tag{10}$$

The extracted components can then be found by performing the eigendecomposition of (10), because the component loadings are the eigenvectors associated with the first *p* highest eigenvalues. Thus, we can compute the V-PCA loadings without actually setting up the huge matrix $\mathbf{X}_{V-PCA}$.

Above, we have seen how to compute the loadings, without setting up the huge matrix $\mathbf{X}_{V-PCA}$, as proposed by Cazes et al. (1997). The component scores have not been computed in this way, although they now could be computed as $\mathbf{A}_{V-PCA} = \mathbf{X}_{V-PCA}\mathbf{B}$, because indeed the obtained loading matrix $\mathbf{B}$ is columnwise orthonormal. However, this would require that we nevertheless use the huge matrix $\mathbf{X}_{V-PCA}$. In practice, however, we do not need the component scores for all vertices, but only, for each observation unit, for each component, the segment covering the

component scores for the associated vertices, given by the formulas (6) and (7). Rather than actually computing all component scores for all vertices, which would require setting up the matrix $\mathbf{X}_{V-PCA}$, we can again use a short-cut offered by Cazes et al. (1997), as follows. Let us first define what we may call the positive and negative component loadings matrices, respectively, $\mathbf{B}^+$ and $\mathbf{B}^-$ with generic elements

$$b_{jk}^{+} = \begin{cases} b_{jk} & \text{if } b_{jk} \geq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

$$b_{jk}^{-} = \begin{cases} b_{jk} & \text{if } b_{jk} \leq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

where b_{jk} gives the loading of variable j on component k. Thus, $\mathbf{B}^+$ (and $\mathbf{B}^-$) contains the non-negative (non-positive) elements of $\mathbf{B}$ whereas the negative (positive) elements of $\mathbf{B}$ are replaced by 0. In matrix notation, the bounds of the component scores matrix are given by

$$\mathbf{A}^- = \mathbf{X}^+\mathbf{B}^- + \mathbf{X}^-\mathbf{B}^+ \tag{13}$$

and

$$\mathbf{A}^+ = \mathbf{X}^+\mathbf{B}^+ + \mathbf{X}^-\mathbf{B}^- . \tag{14}$$

It can be seen that (13) and (14) are easily computed, and do not require setting up the huge matrix $\mathbf{X}_{V-PCA}$.

Thus, it has been seen that, we can get (6) and (7) without explicitly having to compute all the component scores for all the vertices. It follows that this computational approach to V-PCA finds the same component loadings and the same segments for the observation units, as the original computational approach to V-PCA. We only lose the component scores of all individual vertices, but not of the segments that enclose them.

3. C-PCA: Centers Principal Component Analysis

An alternative exploratory tool in order to summarize interval valued data sets is the Centers Principal Component Analysis (C-PCA), as proposed by Cazes et al. (1997). Similarly to V-PCA, C-PCA transforms the interval valued data matrix in (2) into a new single valued matrix. Specifically, the interval valued score of the generic observation unit i on the generic variable j is replaced by the single valued score

$$x_{ij}^{c} = \frac{x_{ij}^{+} + x_{ij}^{-}}{2}, \tag{15}$$

for $i = 1,\ldots,n$ and $j = 1,\ldots,m$, that is the *midpoint* or *center* of the interval at hand. Therefore, we then get the centers matrix

$$\mathbf{X}_{C-PCA} = \begin{pmatrix} x_{11}^c & \cdots & x_{1m}^c \\ \vdots & \ddots & \vdots \\ x_{n1}^c & \cdots & x_{nm}^c \end{pmatrix}. \tag{16}$$

In C-PCA, a PCA is performed on the standardized (in the classical way) matrix in (17).
In order to compare C-PCA to V-PCA, it is interesting to compute the cross-products matrix for the Centers method. Starting from $\mathbf{X}_{C-PCA}$ in (16), we have

$$\mathbf{C}_{C-PCA} = \begin{bmatrix} \sum_{i=1}^{n}\left(x_{i1}^c\right)^2 & \sum_{i=1}^{n} x_{i1}^c x_{i2}^c & \cdots & \sum_{i=1}^{n} x_{i1}^c x_{im}^c \\ \sum_{i=1}^{n} x_{i2}^c x_{i1}^c & \sum_{i=1}^{n}\left(x_{i2}^c\right)^2 & \cdots & \sum_{i=1}^{n} x_{i2}^c x_{im}^c \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} x_{im}^c x_{i1}^c & \sum_{i=1}^{n} x_{im}^c x_{i2}^c & \cdots & \sum_{i=1}^{n}\left(x_{im}^c\right)^2 \end{bmatrix}. \tag{17}$$

By means of (15), (17) can be rewritten as

$$\mathbf{C}_{C-PCA} = \frac{1}{4}\begin{bmatrix} \sum_{i=1}^{n}\left(x_{i1}^- + x_{i1}^+\right)^2 & \sum_{i=1}^{n}\left(x_{i1}^- + x_{i1}^+\right)\left(x_{i2}^- + x_{i2}^+\right) & \cdots & \sum_{i=1}^{n}\left(x_{i1}^- + x_{i1}^+\right)\left(x_{im}^- + x_{im}^+\right) \\ \sum_{i=1}^{n}\left(x_{i2}^- + x_{i2}^+\right)\left(x_{i1}^- + x_{i1}^+\right) & \sum_{i=1}^{n}\left(x_{i2}^- + x_{i2}^+\right)^2 & \cdots & \sum_{i=1}^{n}\left(x_{i2}^- + x_{i2}^+\right)\left(x_{im}^- + x_{im}^+\right) \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n}\left(x_{im}^- + x_{im}^+\right)\left(x_{i1}^- + x_{i1}^+\right) & \sum_{i=1}^{n}\left(x_{im}^- + x_{im}^+\right)\left(x_{i2}^- + x_{i2}^+\right) & \cdots & \sum_{i=1}^{n}\left(x_{im}^- + x_{im}^+\right)^2 \end{bmatrix}. \tag{18}$$

Therefore, by comparing (10) and (18), it is easy to see that the off-diagonal elements of the two cross-products matrices are equal up to a constant 2^m. Ignoring these constants, the matrices differ in the diagonal elements only. Hence, the proportion of elements that differ across the two matrices is $m/m^2 = 1/m$. As a consequence, when the number of variables increases, the cross-products matrices on which the eigendecompositions are performed differ on a smaller proportion of elements. Disregarding the constants in (10) and (18), the difference between each pair of diagonal elements is

$$\sum_{i=1}^{n}\left(x_{ij}^+ - x_{ij}^-\right)^2 = \sum_{i=1}^{n} w_{ij}^2 , \tag{19}$$

$j = 1,\ldots,m$, where w_{ij} denotes the interval width of the *i*-th observation unit, $i = 1,\ldots,n$, with respect to the *j*-th variable, $j = 1,\ldots,m$. Hence, the differences between the matrices depend only on the interval widths, as can be seen in (19).
As for V-PCA, it can be desirable to plot the entities of the unit mode in the obtained low dimensional space. In this case, assuming, without loss of generality, that (16) is already preprocessed, the *k*-th component score of the *i*-th center is

$$a_{ik}^{c} = \sum_{j=1}^{m} x_{ij}^{c} b_{jk} \,, \tag{20}$$

provided that **B** is columnwise orthonormal. It is worthwhile to note that, after preprocessing the observed (lower and upper) bounds using the mean and the standard deviation of the centers of the variable at hand, one may compute the component scores for the vertices (see Cazes et al., 1997). This can be done in the same way as for V-PCA, but then, of course, using as component loadings those obtained performing C-PCA. Again, as for V-PCA, it is attractive to only determine the hyperrectangles that envelop all projected vertices, and for this purpose, we can again use (13) and (14), but then, of course, again using as component loadings those obtained performing C-PCA.
Summing up, the Centers method is nothing but performing ordinary PCA on the midpoints matrix. It follows that C-PCA as such does not exploit all the available information in detecting the underlying structure of the data. To use the interval information, the vertices of all data hyperrectangles are projected on the obtained subspace and next segments enveloping these projections are determined in the same way as for V-PCA, using the (columnwise orthonormal) loadings matrix.

4. NN-PCA: Neural Networks Principal Component Analysis

The PCA extension for fuzzy data proposed by Denœux and Masson (2004) is based upon the connection between classical and autoassociative multilayer perceptrons noticed by Boulard and Kamp (1988). In fact, Boulard and Kamp (1988) showed that standard PCA can be seen as a constrained feedforward three-layer neural network with one hidden layer of size p. Specifically, when $p<m$, the neural network finds the best compression of the input layer (the raw data **X**) according to the quadratic loss function:

$$f(\mathbf{S},\mathbf{T}) = \left\| \mathbf{X} - \hat{\mathbf{X}} \right\|^2 = \left\| \mathbf{X} - \mathbf{XTS} \right\|^2, \tag{21}$$

where the output layer (the estimated data $\hat{\mathbf{X}} = \mathbf{XTS}$) is defined implicitly in (21), **S** ($p \times m$) is the hidden-to-output weights matrix and **T** ($m \times p$) is the input-to-hidden weights matrix. Baldi and Hornik (1989) showed that the global minimum of (21) is given by $\mathbf{S}=\mathbf{U}\mathbf{B}'$ and $\mathbf{T}=\mathbf{B}\mathbf{U}^{-1}$, where **B** ($m \times p$) contains the eigenvectors associated with the p highest eigenvalues of the cross-products matrix $\mathbf{C}=\mathbf{X}'\mathbf{X}$ and **U** ($p \times p$) is an arbitrary nonsingular matrix. From the above description of **S** and **T** it follows that $\hat{\mathbf{X}} = \mathbf{XTS} = \mathbf{XBB}'\mathbf{=AB}'$, where **A** is defined implicitly. Hence, we can conclude that classical PCA can be seen as a particular neural network with one hidden layer of size equal to the number of extracted components. Specifically, the role of the hidden layer is to compress the data at hand as is the role of the components in PCA.

The PCA extension for fuzzy data is based on a suitable distance for comparing observed and estimated fuzzy data. Denœux and Masson (2004) propose to consider the following quadratic loss function:

$$f(\mathbf{B}) = \left\| \left(\mathbf{X}_L^- - \hat{\mathbf{X}}_L^- \right) \right\|^2 + \left\| \left(\mathbf{X}^- - \hat{\mathbf{X}}^- \right) \right\|^2 + \left\| \left(\mathbf{X}^+ - \hat{\mathbf{X}}^+ \right) \right\|^2 + \left\| \left(\mathbf{X}_R^+ - \hat{\mathbf{X}}_R^+ \right) \right\|^2, \qquad (22)$$

which compares all the 'ingredients' of fuzzy intervals. In order to define the component scores matrix and the estimated fuzzy data matrix $\hat{\mathbf{X}} = \left(\hat{\mathbf{X}}_L^-, \hat{\mathbf{X}}^-, \hat{\mathbf{X}}^+, \hat{\mathbf{X}}_R^+ \right)$, Denœux and Masson (2004) make use of the extension principle (see, for instance, Zadeh, 1975). According to the extension principle, the fuzzy component scores matrix can be obtained as

$$\begin{aligned} &\mathbf{A} = \left(\mathbf{A}_L^-, \mathbf{A}^-, \mathbf{A}^+, \mathbf{A}_R^+ \right) = \\ &\left(\mathbf{X}_L^- \mathbf{B}^+ + \mathbf{X}_R^+ \mathbf{B}^-, \mathbf{X}^- \mathbf{B}^+ + \mathbf{X}^+ \mathbf{B}^-, \mathbf{X}^+ \mathbf{B}^+ + \mathbf{X}^- \mathbf{B}^-, \mathbf{X}_R^+ \mathbf{B}^+ + \mathbf{X}_L^- \mathbf{B}^- \right), \end{aligned} \qquad (23)$$

where $\mathbf{B}^+$ and $\mathbf{B}^-$ are constructed according to (11) and (12) but the component loadings matrix is that obtained from NN-PCA. The estimated fuzzy data matrix is then given by

$$\begin{aligned} &\hat{\mathbf{X}} = \left(\hat{\mathbf{X}}_L^-, \hat{\mathbf{X}}^-, \hat{\mathbf{X}}^+, \hat{\mathbf{X}}_R^+ \right) = \\ &\left(\mathbf{A}_L^- \mathbf{B}^{+\prime} + \mathbf{A}_R^+ \mathbf{B}^{-\prime}, \mathbf{A}^- \mathbf{B}^{+\prime} + \mathbf{A}^+ \mathbf{B}^{-\prime}, \mathbf{A}^+ \mathbf{B}^{+\prime} + \mathbf{A}^- \mathbf{B}^{-\prime}, \mathbf{A}_R^+ \mathbf{B}^{+\prime} + \mathbf{A}_L^- \mathbf{B}^{-\prime} \right), \end{aligned} \qquad (24)$$

which, taking into account (23), can be written as

$$\begin{aligned} &\hat{\mathbf{X}} = \left(\hat{\mathbf{X}}_L^-, \hat{\mathbf{X}}^-, \hat{\mathbf{X}}^+, \hat{\mathbf{X}}_R^+ \right) = \\ &\left(\begin{array}{l} \left(\mathbf{X}_L^- \mathbf{B}^+ + \mathbf{X}_R^+ \mathbf{B}^- \right) \mathbf{B}^{+\prime} + \left(\mathbf{X}_R^+ \mathbf{B}^+ + \mathbf{X}_L^- \mathbf{B}^- \right) \mathbf{B}^{-\prime}, \left(\mathbf{X}^- \mathbf{B}^+ + \mathbf{X}^+ \mathbf{B}^- \right) \mathbf{B}^{+\prime} + \left(\mathbf{X}^+ \mathbf{B}^+ + \mathbf{X}^- \mathbf{B}^- \right) \mathbf{B}^{-\prime}, \\ \left(\mathbf{X}^+ \mathbf{B}^+ + \mathbf{X}^- \mathbf{B}^- \right) \mathbf{B}^{+\prime} + \left(\mathbf{X}^- \mathbf{B}^+ + \mathbf{X}^+ \mathbf{B}^- \right) \mathbf{B}^{-\prime}, \left(\mathbf{X}_R^+ \mathbf{B}^+ + \mathbf{X}_L^- \mathbf{B}^- \right) \mathbf{B}^{+\prime} + \left(\mathbf{X}_L^- \mathbf{B}^+ + \mathbf{X}_R^+ \mathbf{B}^- \right) \mathbf{B}^{-\prime} \end{array} \right). \end{aligned} \qquad (25)$$

Therefore, (22) can be elaborated as

$$\begin{aligned} f(\mathbf{B}) = &\left\| \left(\mathbf{X}_L^- - \left(\mathbf{X}_L^- \mathbf{B}^+ + \mathbf{X}_R^+ \mathbf{B}^- \right) \mathbf{B}^{+\prime} + \left(\mathbf{X}_R^+ \mathbf{B}^+ + \mathbf{X}_L^- \mathbf{B}^- \right) \mathbf{B}^{-\prime} \right) \right\|^2 + \\ &\left\| \left(\mathbf{X}^- - \left(\mathbf{X}^- \mathbf{B}^+ + \mathbf{X}^+ \mathbf{B}^- \right) \mathbf{B}^{+\prime} + \left(\mathbf{X}^+ \mathbf{B}^+ + \mathbf{X}^- \mathbf{B}^- \right) \mathbf{B}^{-\prime} \right) \right\|^2 + \\ &\left\| \left(\mathbf{X}^+ - \left(\mathbf{X}^+ \mathbf{B}^+ + \mathbf{X}^- \mathbf{B}^- \right) \mathbf{B}^{+\prime} + \left(\mathbf{X}^- \mathbf{B}^+ + \mathbf{X}^+ \mathbf{B}^- \right) \mathbf{B}^{-\prime} \right) \right\|^2 + \\ &\left\| \left(\mathbf{X}_R^+ - \left(\mathbf{X}_R^+ \mathbf{B}^+ + \mathbf{X}_L^- \mathbf{B}^- \right) \mathbf{B}^{+\prime} + \left(\mathbf{X}_L^- \mathbf{B}^+ + \mathbf{X}_R^+ \mathbf{B}^- \right) \mathbf{B}^{-\prime} \right) \right\|^2. \end{aligned} \qquad (26)$$

The minimization of (26) is attained by means of a standard gradient descent procedure. As far as we can see, such a minimization procedure does not constrain the loadings to be columnwise orthonormal. See, for more details, Denœux and Masson (2004).

As with C-PCA and V-PCA, the component scores can be used to plot (hyper)rectangles for the objects. In the case of NN-PCAS, this is particularly easy, because for each object we have fuzzy interval component scores, so we can use the supports, or the cores for each component to define the (hyper)rectangles to be plotted.

In conclusion, like V-PCA and C-PCA, NN-PCA finds a crisp loading matrix, but in contrast to V-PCA and C-PCA, NN-PCA finds fuzzy interval component scores exploiting the potential of autoassociative multilayer neural networks and taking into account the extension principle.

Before presenting the results of the simulation study and of the application, it is important to note that C-PCA, as PCA, finds the best rank *p* decomposition of the data matrix at hand (the centers matrix). If such a matrix has rank *p*, it follows that C-PCA perfectly decomposes it into two rank *p* matrices, namely the component scores and the loadings ones and such a decomposition gives exactly the original data matrix. However, C-PCA is not able to optimally capture the data uncertainty. In fact, C-PCA only models the centers by PCA and then in a second step projects the uncertainties on that space, but that space need not optimally capture the uncertainties. Therefore, if the centers matrix has rank *p*, C-PCA can only find *p*-dimensional spaces such that the *m*-dimensional centers perfectly lie in it. Also V-PCA is not able to capture the uncertainties. Specifically, a perfect V-PCA solution would imply a perfect fit of all the vertices, which is impossible, as follows from the fact that the vertices matrix in (5) with $n2^m$ rows and *m* columns cannot lie in the low dimensional space spanned by the ($p < m$) columns of the component loadings matrix. In fact, this would require that the vertices matrix $\mathbf{X}_{V-PCA}$ would have rank *p*, which is obviously not true (due to the structure of such a matrix). Note that this holds whenever the number of extracted components is lower than that of the variables. Basically, there is a geometrical reason for which the obtained low dimensional space does not optimally capture the uncertainty about the data (as given by the intervals). In fact, it is trivial that *m*-dimensional hyperrectangles cannot be fully described in the low dimensional space. If $m = 3$ and $p = 2$, we could never construct three-dimensional data (hyperrectangles) in such a way that all data lie in a plane. Similar comments hold concerning NN-PCA. That is, it seems hard, if not impossible, to find nontrivial data matrices **X** and component loadings matrices **B** of rank *p*, for which the distance in (26) is 0.

Therefore, both C-PCA and V-PCA, which involve the use of standard mathematical tools, and NN-PCA, which involves the use of the extension principle and the concepts derived from fuzzy arithmetic cannot perfectly capture the uncertainty of the data. On the contrary, in classical PCA, we are able to construct *m*-dimensional data (points) that lie in a *p*-dimensional subspace.

5. Simulation study

In the previous sections, we reviewed three methods for performing a component analysis on fuzzy data. In C-PCA and V-PCA, the methods suitably transform the

interval valued data matrix into a single valued one on which classical PCA is performed. We also showed that the cross-products matrices obtained considering C-PCA and V-PCA differ only in considering the interval width in the diagonal elements to find the underlying components. Thus, it has become clear that C-PCA and V-PCA are rather similar. Instead, in NN-PCA, computation of component scores and data estimates is based on the extension principle, which is a quite different approach. To choose, among the methods, it is now necessary to know which method performs best in practice. To gain knowledge on the performance of the three methods, we set up a simulation study in which fuzzy interval data are generated according to a model, and next C-PCA, V-PCA and NN-PCA are applied to these data, and it is inspected to what extent each method recovers the underlying information. Specifically, we analyzed the performance of the methods with respect to recovering the known component loadings and the component scores. Note that C-PCA and V-PCA are applied to the cores of the thus constructed fuzzy interval data.

We simulated data sets with fuzzy intervals for n (which was chosen equal to 16, 32, 48 or 64) observation units and m (which was chosen equal to 6, 12, 18 or 24) variables, as follows. We constructed a component loadings matrix (**B**) with p=2 or 3 components. Specifically, we considered two situations. In the first one, all the elements were randomly generated from the uniform distribution on $[0,1]$ or the standard normal distribution and the matrix was columnwise orthonormalized. In the latter one, **B** was constructed so that it had a simple structure. To do so, **B** was built as

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{b}_2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{b}_p \end{bmatrix}, \tag{27}$$

where **0**'s are m/p-vectors with zero elements, and $\mathbf{b}_k$, $k=1,\ldots,p$ are m/p-vectors with elements equidistantly chosen from the interval [0.6, 0.9]; for instance, if m=12 and p=3, we got $\mathbf{b}_k = [0.6 \quad 0.7 \quad 0.8 \quad 0.9]'$, $k=1,\ldots,p$. Then **B** was orthonormalized: this was done by multiplying **B** by $(\mathbf{B}'\mathbf{B})^{-1/2}$. Next, we constructed a component scores matrix of fuzzy intervals for p (=2 or 3) components. Specifically, for each observation unit, we generated the lower bounds of the fuzzy intervals (from the uniform distribution on $[0,1]$ or from the standard normal distribution) and the width of such fuzzy intervals considering the following three choices. In the *small* width case, we have widths equal to 0.4 for the first component, 0.3 for the second one and 0.2 for the third one if p=3; in the *medium* case, 0.8, 0.7 and 0.6 (the last only if p=3); in the *large* case, 1.2, 1.1 and 1.0 (the last only if p=3). By adding the width to the lower bounds, we obtained the upper bounds. The upper and lower bounds (hence, the support) of the fuzzy component scores for the i-th observation unit, $i=1,\ldots,n$, are denoted as a^{-}_{Lik} and a^{+}_{Rik}, $k=1,\ldots,p$. Similarly, we constructed the core of the fuzzy component scores. This was done according to four assumptions concerning the left and right spreads of the component scores which are constructed as a percentage (20% or 40%) of the support of the fuzzy component scores going from a^{-}_{Lik} to a^{+}_{Rik}, $i=1,\ldots,n$, $k=1,\ldots,p$. This allowed us to check how the methods worked in case of fuzzy component scores with small (20%) and big (40%) symmetric spreads and in case of fuzzy component scores with asymmetric spreads

with small or big cores. Next, given a component score matrix with fuzzy intervals $\mathbf{A} = \left(\mathbf{A}_L^-, \mathbf{A}^-, \mathbf{A}^+, \mathbf{A}_R^+\right)$ and the component loading matrix **B**, we constructed our fuzzy interval data (i.e., their supports and cores) according to (24). Let $\mathbf{X} = \left(\mathbf{X}_L^-, \mathbf{X}^-, \mathbf{X}^+, \mathbf{X}_R^+\right)$ be the resulting matrix of fuzzy intervals. Then, noise was added to this matrix. The noise values were generated from the standard normal distribution, when the component scores were based on the normal distribution, and from the uniform on $[-0.5, 0.5]$, when the component scores were based on the uniform distribution. In particular, we considered five different levels of added noise (which was chosen as $e = 0.01, 0.10, 0.50, 1.00, 2.00$). Let $\mathbf{E}_{X_L^-}$, $\mathbf{E}_{X^-}$, $\mathbf{E}_{X^+}$ and $\mathbf{E}_{X_R^+}$ denote the matrices containing noise values for $\mathbf{X}_L^-$, $\mathbf{X}^-$, $\mathbf{X}^+$ and $\mathbf{X}_R^+$, respectively. These matrices were next scaled such that $\left\|\mathbf{E}_{X_L^-}\right\|^2 = \left\|\mathbf{X}_L^-\right\|^2$, $\left\|\mathbf{E}_{X^-}\right\|^2 = \left\|\mathbf{X}^-\right\|^2$, $\left\|\mathbf{E}_{X^{+-}}\right\|^2 = \left\|\mathbf{X}^+\right\|^2$ and $\left\|\mathbf{E}_{X_R^+}\right\|^2 = \left\|\mathbf{X}_R^+\right\|^2$, in order to quantify exactly the relative amount of noise added to $\mathbf{X}_L^-$, $\mathbf{X}^-$, $\mathbf{X}^+$ and $\mathbf{X}_R^+$, which was, respectively, $e\mathbf{E}_{X_L^-}$, $e\mathbf{E}_{X^-}$, $e\mathbf{E}_{X^+}$ and $e\mathbf{E}_{X_R^+}$. If, after adding noise to the fuzzy data matrix **X**, it happened that, for some elements, the lower bound $\mathbf{X}_L^-$ and $\mathbf{X}_R^+$, any element of $\mathbf{X}_L^-$ was higher than the corresponding one of $\mathbf{X}_R^+$, we replaced the values involved by the average one, hence leading to a crisp number. Note that the midpoints matrix on which C-PCA is performed was obtained as

$$\mathbf{X}_{C-PCA} = \frac{\mathbf{X}^+ + \mathbf{X}^-}{2}. \tag{28}$$

In the sequel, we refer to the case in which **B** and **A** are generated from the normal (uniform) distribution as the *normal* (*uniform*) case.

The aim of the simulation study is to study which method works best not only in recovering the known component loadings but also the component scores. For each combination of number of observation units (16, 32, 48, 64), number of variables (6, 12, 18, 24), number of extracted components (2, 3), interval width (small, medium, large), level of added noise ($e = 0.01, 0.10, 0.50, 1.00, 2.00$), structure of the left and right spreads (both 20%, 20% and 40%, 40% and 20%, both 40%), structure of **B** (random or simple) and generating distribution (normal or uniform), 10 data sets were generated. Thus, the simulation was carried out on 4×4×2×3×5×4×2×2×5 = 38400 fuzzy data sets.

In order to compare the performances of all methods, we must take into account the possible rotational freedom of V-PCA, C-PCA and NN-PCA. Thus, a component loadings matrix obtained with either method must be rotated so that it is as similar as possible to the known one. For instance, with respect to the loadings, if $\mathbf{B}_C$ is the component loadings matrix obtained by means of C-PCA, we rotated $\mathbf{B}_C$ by means of the rotation matrix $\mathbf{T}_C$ in such a way that

$$\left\|\mathbf{B}_C\mathbf{T}_C - \mathbf{B}\right\|^2 = \min_{\mathbf{T}}\left\|\mathbf{B}_C\mathbf{T} - \mathbf{B}\right\|^2. \tag{29}$$

Note that the minimum of (29) can be attained by solving an ordinary regression problem with respect to **T**. The same can be done with respect to the component loadings matrix $\mathbf{B}_V$ obtained performing V-PCA, and for the matrix $\mathbf{B}_{NN}$.obtained with NN-PCA. As far as we can see, the NN-PCA procedure, however, does not leave rotational freedom for **B**. This implies that the results we report for NN-PCA may be more optimistic then in the actual practical use of NN-PCA.

Let $\mathbf{B}_C$, $\mathbf{B}_V$ and $\mathbf{B}_{NN}$ now denote the rotated component loadings matrices. The recovery of the loadings by each method was assessed by means of the Mean Absolute Difference (MAD) between every element of the known component loadings matrix and the estimated one (see, e.g., Giordani and Kiers, 2004) divided by the number of elements of the matrix involved. For the component loadings matrix from C-PCA (similarly for V-PCA and NN-PCA), we have:

$$MAD_{B_C} = \frac{\sum_{j=1}^{m}\sum_{k=1}^{p}\left|b_{C\,jk} - b_{jk}\right|}{mp}, \tag{30}$$

where $b_{C\,jp}$ and b_{jp}, $j = 1,\ldots,m$, $k = 1,\ldots,p$, denote the generic elements of $\mathbf{B}_C$ and **B**, respectively. The closer the MAD value is to 0, the better the method works.

In order to evaluate the recovery of the component scores, we consider the average MAD measure between the known upper and lower bounds of the component scores corresponding to the cores and those obtained by C-PCA, V-PCA and NN-PCA. Note that, for computing these bounds using (13) and (14), we must now use the component weights defined as $\mathbf{B}_C\left(\mathbf{B}_C'\mathbf{B}_C\right)^{-1}$, $\mathbf{B}_V\left(\mathbf{B}_V'\mathbf{B}_V\right)^{-1}$ and $\mathbf{B}_{NN}\left(\mathbf{B}_{NN}'\mathbf{B}_{NN}\right)^{-1}$, respectively. This is because, in general, in ordinary PCA, component scores are given by $\mathbf{A}=\mathbf{XB}(\mathbf{B}'\mathbf{B})^{-1}$, with **A** denoting the component scores matrix, and **B** the loadings matrix, and when **B** is not columnwise orthonormal, this expression cannot be simplified further, so in that case the component *weights* are given by $\mathbf{B}(\mathbf{B}'\mathbf{B})^{-1}$. Due to the (oblique) rotation, the present loading matrices are no longer columwise orthonormal, so the component weights matrices used in (13) and (14) must be based on $\mathbf{B}(\mathbf{B}'\mathbf{B})^{-1}$, where now **B** is replaced by $\mathbf{B}_C$, $\mathbf{B}_V$ or $\mathbf{B}_{NN}$.

Given the component scores, we compute the overall component scores recovery value (for C-PCA, but likewise for V-PCA and NN-PCA) as

$$MAD_{A_C} = \frac{MAD_{A_C^+} + MAD_{A_C^-}}{2}, \tag{31}$$

where, for instance, $MAD_{A_C^+} = \dfrac{\sum_{i=1}^{n}\sum_{k=1}^{p}\left|a_{C\,ik}^{+} - a_{ik}^{+}\right|}{np}$ in which $a_{C\,ik}^{+}$ and a_{ik}^{+} are the generic elements of the matrices of, respectively, the obtained upper bounds of the cores of the component score intervals and of the known ones.

In a few cases, we observed that the recovery of the known loadings was quite bad and the best-rotated component loadings matrices were nearly collinear. If **B** is nearly collinear, the weights in **W** become very large. As a matter of fact, this causes a very poor recovery of the component scores. The frequency (out of 9600 cases) of such

extreme cases is given in Table 1.

Table 1: *Frequency (out of 9600 cases) of extremely poor recovery of the component scores: MAD values higher than 10, 25 and 50 distinguished with respect to the method at hand, the generating distribution and the structure of the component loadings matrix.*

MAD	Uniform Case Random Loadings			Uniform Case Simple Loadings			Normal Case Random Loadings			Normal Case Simple Loadings		
	C-	V-	NN-	C-	V-	NN-	C-	V-	NN-	C-	V-	NN-
>10	134	146	187	114	152	224	10	16	27	5	3	16
>25	54	58	67	35	61	91	6	7	11	0	1	5
>50	25	27	37	16	32	45	3	2	6	0	1	1

We noted that this misrecovery often occurred when the level of noise was high ($e \geq 1.0$) and especially when $e = 2.0$.

In order to avoid that such anomalous cases will dominate our conclusions, in comparing the methods, we considered the median MAD values. The results are given in Figures 2 and 3. All the figures display the median MAD values distinguished with respect to the number of observation units, number of variables, number of extracted components, level of added noise, the structure of the fuzzy intervals (symmetry and size of the spreads) and the choice of the component scores widths. In particular, Figure 2 refers to the recovery of the known component loadings according to (30) and Figure 3 to that of the known component scores according to (31).

Figures 2 and 3 provide useful information in order to assess which method works best in which situation. With respect to the recovery of the loadings, we observe that, in the *uniform* case, the results were conflicting. Specifically, in the simple structure case, NN-PCA should be preferred, even when the level of added noise increased (e=1.0 or e=2.0), the performance of NN-PCA got worse. On the contrary, when the loadings were randomly generated, C-PCA worked somewhat better than V-PCA and, above all, than NN-PCA. In the *normal* case, when the loadings had simple structure, NN-PCA worked well and better than C-PCA and V-PCA. When the loadings were randomly generated, V-PCA and C-PCA seemed to work equally well and much better than NN-PCA. As far as the recovery of the component scores is concerned (Figure 3), in the *uniform* case with simple structure loadings, all the three methods appeared to work equally well, with NN-PCA performing relatively poorly in the highest noise condition. When the loadings had random structure, we observed that NN-PCA performed considerably more poorly than C-PCA and V-PCA. In the *normal* case, all the three methods worked almost similarly.

Figure 2: *Badness of recovery of component loadings (median MAD values).*

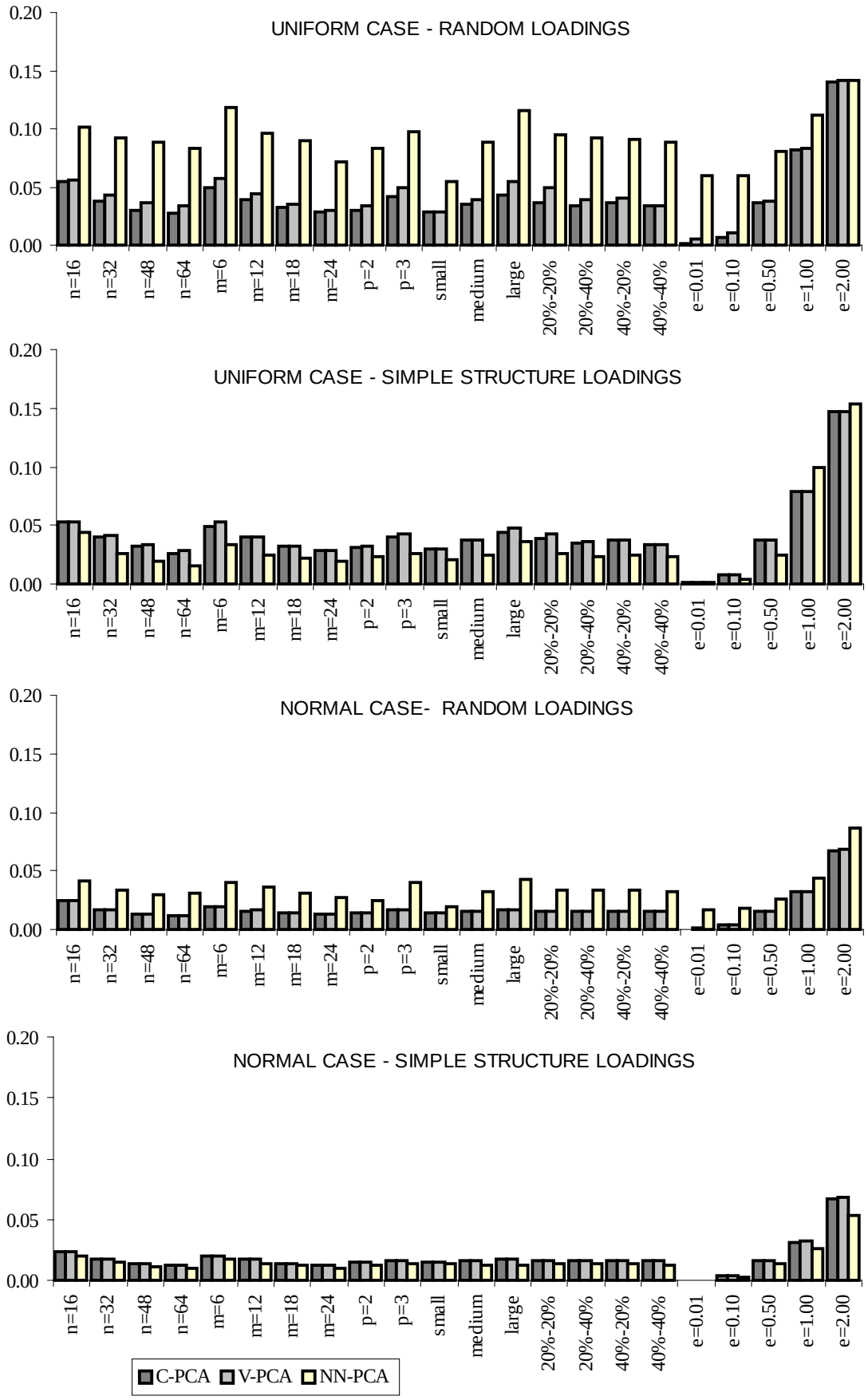
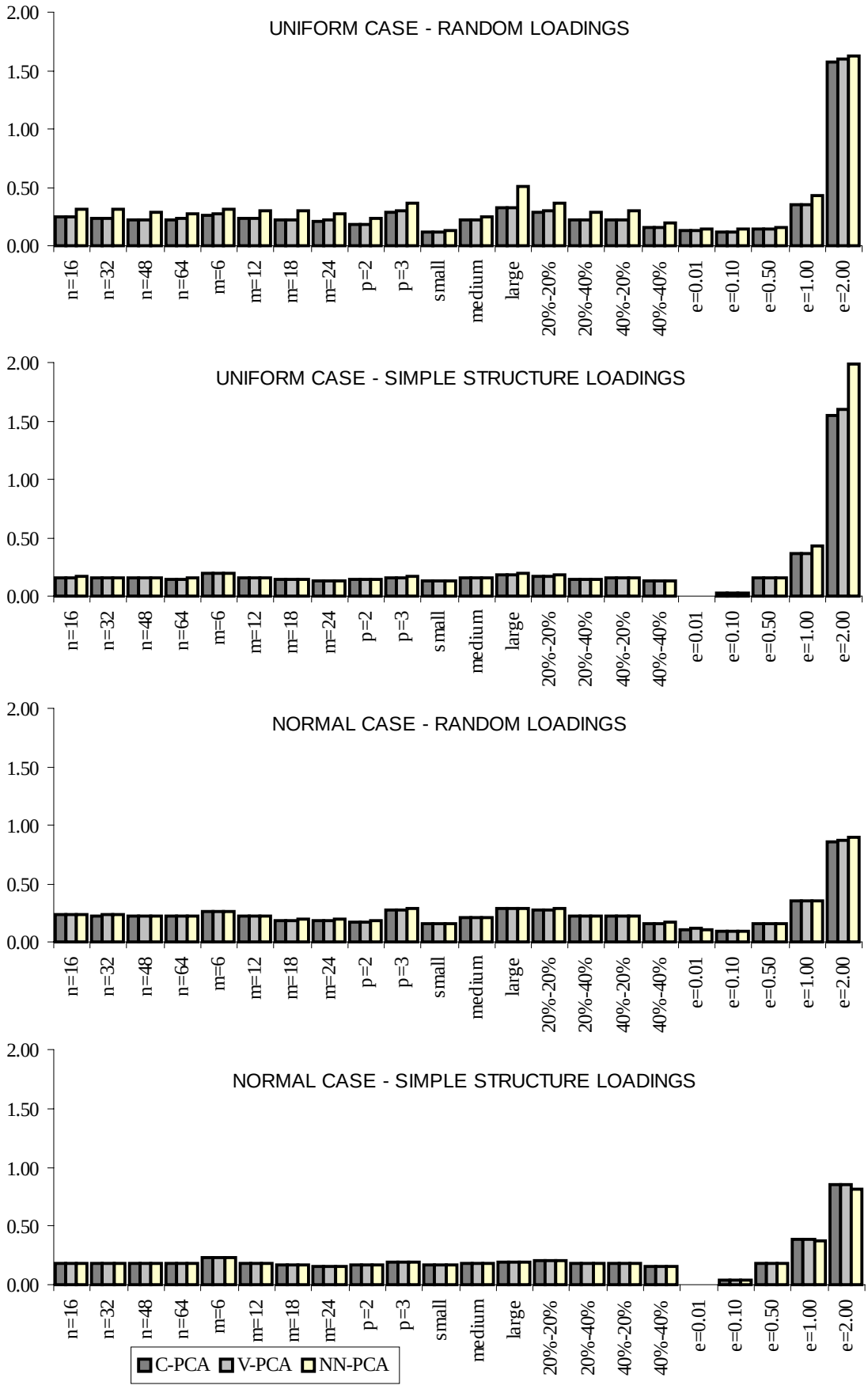


Figure 3: *Badness of recovery of component scores (median MAD values).*



Further details can be found in Tables 2 and 3 in which we summarize how many times one method worked noticeably better than the others in recovering the known component loadings (Table 2) and scores (Table 3). As Figures 2 and 3 showed that C-PCA and V-PCA worked almost similarly (with C-PCA working uniformly slightly better than V-PCA), we decided to limit our comparison to C-PCA and NN-PCA.
In Tables 2 and 3, each cell gives the percentage of the number of times in which the method in the column worked noticeably better than the other method, both overall, and in each the conditions mentioned in the first column. The conditions are the number of observation units, the number of variables, the number of extracted components, the interval width for the component scores, the structure of the fuzzy intervals (symmetry and size of the spreads) and the level of added noise. Here we decided to consider that one method performed noticeably better than the other one when the difference on the MAD index exceeded a threshold value equal to 0.03 for the recovery of the component loadings and 0.30 for the one of the component scores. If the difference was smaller, we considered the difference to be negligible.

Table 2: *Percentage of the number of times in which one method worked noticeably better with respect to the recovery of the component loadings.*

Condition	Uniform Case Random Loadings		Uniform Case Simple Loadings		Normal Case Random Loadings		Normal Case Simple Loadings	
	C-	NN-	C-	NN-	C-	NN-	C-	NN-
Overall	53.1	3.3	12.2	5.2	16.8	0.5	1.3	2.4
n=16	44.8	3.7	12.6	4.3	15.5	1.5	0.9	4.4
n=32	52.2	3.8	11.0	5.9	18.0	0.4	2.2	3.0
n=48	55.9	2.8	12.4	6.4	15.0	0.0	1.6	1.2
n=64	60.6	3.1	13.3	4.3	19.3	0.0	0.6	1.3
m=6	57.7	10.5	18.5	12.3	30.3	1.7	3.9	7.5
m=12	55.5	2.5	11.3	5.4	19.3	0.2	0.6	1.6
m=18	53.6	0.1	10.3	2.3	10.6	0.0	0.5	0.6
m=24	46.8	0.2	9.2	0.9	7.5	0.0	0.2	0.1
p=2	47.3	2.8	11.8	4.5	6.5	0.6	1.1	2.1
p=3	59.0	3.9	12.5	5.9	27.1	0.4	1.6	2.8
small	35.4	2.5	6.8	5.4	4.2	0.3	0.7	2.0
medium	60.3	3.0	12.5	5.2	15.0	0.5	1.2	2.6
large	64.5	4.5	17.8	5.1	31.6	0.7	2.1	2.7
20-20	52.5	3.6	12.0	5.4	17.6	0.5	1.2	2.7
20-40	54.1	3.2	12.4	4.9	17.1	0.5	1.2	2.5
40-20	52.9	3.4	12.0	5.7	16.5	0.5	1.5	2.4
40-40	54.0	3.1	13.0	4.8	16.5	0.5	1.4	2.3
e=0.01	84.6	0.0	1.7	0.0	25.1	0.0	0.0	0.0
e=0.10	79.9	0.0	1.6	0.0	20.2	0.0	0.0	0.0
e=0.50	55.7	0.5	11.5	7.2	8.2	0.0	0.0	0.2
e=1.00	33.4	7.2	30.2	9.9	8.8	1.0	0.7	1.6
e=2.00	13.3	8.9	16.7	8.9	22.3	1.5	5.9	10.5

Concerning the recovery of the loadings (Table 2), in the *uniform* case, C-PCA relatively often performed noticeably better than NN-PCA, while the reverse did not happened often. This was found in the *simple structure* case, and more strongly in the

random case. In the *normal* case with random loadings, again C-PCA relatively often performed noticeably better than NN-PCA, while now the reverse happened only rarely. In the *normal case* with simple loadings, neither method often performed noticeably better than the other, but as far as this happened, it happened somewhat more often for NN-PCA than for C-PCA.
As far as the recovery of the component scores is concerned, Table 3 shows that, in the *normal* case, neither method often performed noticeably better than the other, for both random and simple structure loadings. However, in the few cases where there are noticeable differences, these are found more often in favor of C-PCA than of NN-PCA. Conversely, in the *uniform* case, the C-PCA method relatively frequently performed noticeably better than NN-PCA, while the reverse happened considerably less often.

Table 3: *Percentages of the number of times in which one method worked noticeably better with respect to the recovery of the component scores.*

Condition	Uniform Case Random Loadings		Uniform Case Simple Loadings		Normal Case Random Loadings		Normal Case Simple Loadings	
	C-	NN-	C-	NN-	C-	NN-	C-	NN-
Overall	16.4	7.6	15.0	5.9	1.8	0.7	1.7	0.4
n=16	18.6	8.3	17.7	7.9	2.8	2.0	3.3	1.5
n=32	15.3	8.8	13.0	6.6	2.1	0.2	2.2	0.1
n=48	16.6	6.8	15.3	6.0	1.1	0.5	0.9	0.0
n=64	15.3	6.5	14.3	3.1	1.3	0.0	0.6	0.1
m=6	13.3	8.5	14.0	9.8	4.4	1.6	3.9	0.3
m=12	15.5	9.0	14.4	5.0	1.8	0.6	0.6	0.7
m=18	21.0	6.1	15.0	4.9	0.9	0.5	0.8	0.7
m=24	16.0	6.9	17.0	3.9	0.2	0.0	1.6	0.0
p=2	15.1	5.4	11.8	4.7	0.8	0.1	0.4	0.0
p=3	17.6	9.9	18.2	7.1	2.8	1.3	3.0	0.8
small	6.9	4.5	9.5	3.7	1.1	0.4	1.3	0.3
medium	14.2	7.9	15.6	5.8	1.7	0.7	1.7	0.4
large	28.3	10.5	20.2	8.3	2.6	1.0	2.3	0.6
20-20	17.6	8.3	15.3	6.8	1.8	0.7	1.6	0.4
20-40	17.0	7.3	14.8	5.6	1.6	0.6	1.6	0.5
40-20	17.0	7.7	15.0	6.1	1.9	0.8	1.9	0.4
40-40	14.2	7.2	15.2	5.3	2.0	0.6	1.9	0.4
e=0.01	9.6	0.0	1.1	0.0	0.0	0.0	0.0	0.0
e=0.10	8.4	0.0	1.0	0.0	0.0	0.0	0.0	0.0
e=0.50	8.2	0.4	5.1	0.1	0.0	0.0	0.0	0.0
e=1.00	11.5	5.9	21.3	4.1	0.2	0.0	0.6	0.0
e=2.00	23.6	31.8	46.9	25.5	8.9	3.4	8.1	2.1

Summing up, the results of the simulation study showed that for the uniform case C-PCA had the best overall performance (especially for the *random* loadings case). Such a comment does not hold in the *normal* case, where NN-PCA performed best in the *simple* loadings case and C-PCA in the *random* case. On the whole, NN-PCA seemed to work better in the *simple* loadings case, as compared to the *random* case. Finally, the performance of the three methods could be ordered as C-PCA, NN-PCA and V-PCA.

6. Application

In this section we illustrate the results of an application of the three methods to a fuzzy interval data set describing 16 fruit juices evaluated by a group of judges on 6 features. More specifically, there are eight fruit juices (apple, apricot, banana, pineapple, grapefruit, orange, peach and peer) and two brands for each juice. The features are appearance, smell, naturalness, taste, density and sweetness. All the judges evaluated the appearance and the smell before tasting and the remaining characteristics later. To evaluate each attribute, a scale, whose values are from 1 to 10, is used. Unfortunately, the interindividual differences in judges were unknown.

Table 4. *Fruit juices data.*

Fruit juices	Appearance	Smell	Taste	Naturalness	Sweetness	Density
Apple1	(6.78,6.78, 7.50,7.52)	(5.47,5.59, 6.49,6.59)	(7.40,7.40, 8.17,8.40)	(5.66,5.77, 6.86,7.20)	(7.27,7.27, 7.99,8.29)	(5.81,5.81, 6.7,6.74)
Apple2	(6.60,6.79, 7.64,7.72)	(6.28,6.34, 7.23,7.40)	(6.31,6.32, 7.33,7.43)	(5.72,5.87, 6.91,7.12)	(6.67,6.67, 7.57,7.65)	(5.47,5.55, 6.53,6.59)
Apricot1	(6.82,6.82, 7.50,7.68)	(7.87,7.87, 8.45,8.68)	(7.60,7.60, 8.36,8.54)	(7.35,7.51, 8.25,8.47)	(7.42,7.46, 8.11,8.40)	(7.03,7.04, 7.82,8.15)
Apricot2	(7.32,7.53, 8.15,8.16)	(7.09,7.09, 7.89,8.19)	(5.17,5.42, 6.42,6.71)	(4.66,4.81, 5.82,6.06)	(4.90,5.15, 6.15,6.31)	(5.79,5.87, 6.72,6.77)
Banana1	(4.96,5.24, 6.21,6.37)	(3.92,4.14, 5.20,5.60)	(3.64,4.13, 5.20,5.32)	(4.27,4.63, 5.68,5.95)	(4.76,4.98, 5.92,6.16)	(3.62,3.78, 4.73,4.74)
Banana2	(5.27,5.46, 6.46,6.67)	(3.68,3.98, 5.08,5.36)	(3.26,3.58, 4.69,4.94)	(3.92,4.15, 5.18,5.46)	(4.23,4.57, 5.63,5.91)	(3.65,3.83, 4.77,4.77)
Grapefruit1	(6.28,6.30, 7.26,7.40)	(6.52,6.65, 7.59,7.65)	(5.17,5.46, 6.58,6.85)	(6.00,6.16, 7.20,7.33)	(2.45,2.65, 3.39,3.39)	(3.64,3.84, 4.72,4.76)
Grapefruit2	(6.31,6.42, 7.21,7.43)	(5.63,5.83, 6.70,6.75)	(6.35,6.46, 7.30,7.47)	(6.11,6.12, 6.96,7.23)	(4.14,4.14, 5.02,5.19)	(3.06,3.38, 4.34,4.46)
Orange1	(6.64,6.64, 7.44,7.59)	(7.12,7.15, 7.97,8.24)	(6.39,6.39, 7.29,7.44)	(5.67,5.74, 6.70,6.72)	(5.75,5.75, 6.57,6.67)	(3.64,3.80, 4.76,4.97)
Orange2	(6.89,6.93, 7.55,7.55)	(6.06,6.09, 6.87,6.90)	(6.82,6.82, 7.66,7.94)	(5.60,5.75, 6.69,6.72)	(5.93,5.93, 6.89,7.13)	(3.88,4.06, 4.98,4.98)
Peach1	(7.09,7.21, 7.81,7.93)	(6.94,6.94, 7.69,7.78)	(6.42,6.52, 7.44,7.54)	(5.70,5.89, 6.86,7.10)	(6.69,6.75, 7.56,7.68)	(5.03,5.03, 5.92,5.92)
Peach2	(6.98,7.01, 7.74,7.82)	(6.22,6.29, 7.11,7.11)	(7.38,7.38, 8.15,8.38)	(6.83,6.83, 7.60,7.72)	(6.83,6.96, 7.74,7.81)	(4.99,4.99, 5.83,5.85)
Peer1	(6.89,6.89, 7.67,7.76)	(7.19,7.28, 8.04,8.24)	(7.14,7.17, 7.99,8.19)	(6.44,6.47, 7.33,7.49)	(7.59,7.59, 8.37,8.54)	(7.22,7.34, 8.06,8.27)
Peer2	(7.52,7.52, 8.20,8.20)	(6.32,6.40, 7.28,7.44)	(7.69,7.69, 8.33,8.57)	(6.72,6.72, 7.48,7.63)	(7.71,7.71, 8.45,8.62)	(6.72,6.72, 7.60,7.67)
Pineapple1	(6.61,6.77, 7.51,7.66)	(5.74,5.74, 6.54,6.66)	(6.18,6.21, 7.10,7.31)	(5.45,5.52, 6.52,6.85)	(5.63,5.82, 6.71,6.75)	(3.92,4.16, 5.00,5.00)
Pineapple2	(6.66,6.66, 7.42,7.59)	(5.90,6.19, 7.09,7.30)	(5.65,5.84, 6.76,6.98)	(5.23,5.52, 6.48,6.56)	(5.52,5.62, 6.62,6.92)	(3.28,3.69, 4.67,4.69)

In fact, with respect to the rating pertaining to the *i*-th juice and the *j*-th variable, we only know the lower bound (x_L^-), the upper bound (x_R^+), the mean value (m) and the standard deviation (s). Every fuzzy interval datum is then constructed as $(x_L^-, m-s, m+s, x_R^+)$ under the assumption that the data are concentrated near the mean and their likelihood decreases as they are farther from the centers. The way of setting up fuzzy intervals by using simple statistics for a distribution of crisp data is similar to that by, for instance, Denœux and Masson (2004). The data are summarized in Table 4.

On the preprocessed data (by centering them using, for each variable, the mean of the lower and upper bounds), C-PCA, V-PCA and NN-PCA were applied. In particular, C-PCA and V-PCA were applied to the data set given by the cores of the fuzzy intervals, ignoring the spread information. Instead, NN-PCA was applied to the entire fuzzy intervals. The fit values of all the three methods are displayed in Table 5.

Table 5: *Fit values of C-PCA, V-PCA and NN-PCA with different numbers of components.*

Number of components	C-PCA	V-PCA	NN-PCA
p=1	69.71	58.15	71.29
p=2	85.54	73.58	85.34
p=3	93.96	84.29	91.91
p=4	98.76	91.25	97.25
p=5	99.62	96.43	99.04
p=6	100.00	100.00	100.00

Note: The fit value of NN-PCA is obtained dividing the sum of explained squares by the total sum of squares considering the bounds of the core and the left and right spreads.

For all the methods, we decided to extract p=2 components because the fit increase in going from one to two components was still worthwhile, and because in this way all variables were well captured by at least one component. The component loadings matrices (by extracting two components) are given in Table 6. Those resulting from C-PCA and V-PCA are varimax rotated, whereas the loadings from NN-PCA are the optimal ones obtained performing the minimization procedure by Denœux and Masson (2004), without further rotations.

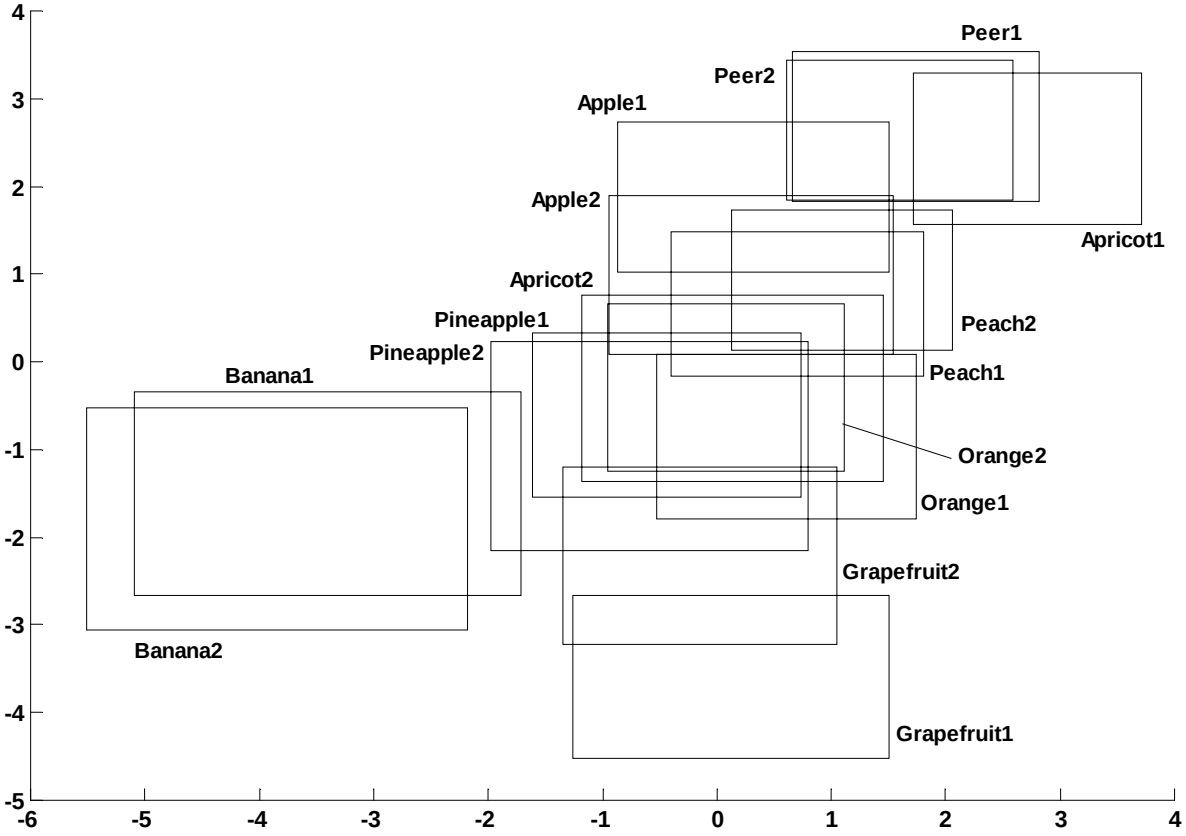
Table 6: *Component loadings matrices from C-PCA, V-PCA and NN-PCA.*

Variables	PC1			PC2		
	C-	V-	NN-	C-	V-	NN-
Appearance	**0.25**	**0.25**	**0.34**	0.04	0.02	-0.03
Smell	**0.65**	**0.61**	**0.51**	-0.14	-0.13	-0.01
Taste	**0.54**	**0.57**	**0.59**	0.14	0.12	0.06
Naturalness	**0.46**	**0.47**	**0.50**	-0.02	-0.04	-0.04
Sweetness	-0.08	-0.07	-0.00	**0.80**	**0.80**	**0.73**
Density	0.13	0.14	0.02	**0.57**	**0.57**	**0.66**

From Table 6, we can observe that the components have similar interpretations. In fact, the first component is especially related to smell, taste and naturalness. These features play a relevant role in evaluating the fruit juices ratings. The appearance is correlated to the overall rating, but it is less important than the other attributes. In fact, it is intuitively plausible that the sense of smell and the palate liking are probably more relevant than the appearance. A fruit juice can be very nice but its appearance may be unpleasant. The second component mainly refers to sweetness and density. These aspects, apparently, are distinghuished clearly from the other four attributes, and is somewhat less related to how much one likes the juices. Therefore, the first component is interpreted as the overall liking. High component scores mean that the fruit juices involved are considered more pleasant. The second component is simply denoted as ‘sweetness and density’.
Using the component score intervals as computed by means of (13) and (14), in Figure 4 we plotted the fruit juices (as rectangles) based on the C-PCA solution - the

differences with the V-PCA solution. In Figure 5, we also plotted the fruit juices using the fuzzy component scores resulting from NN-PCA. It can be seen at once that differences between the two plots are only in the details.

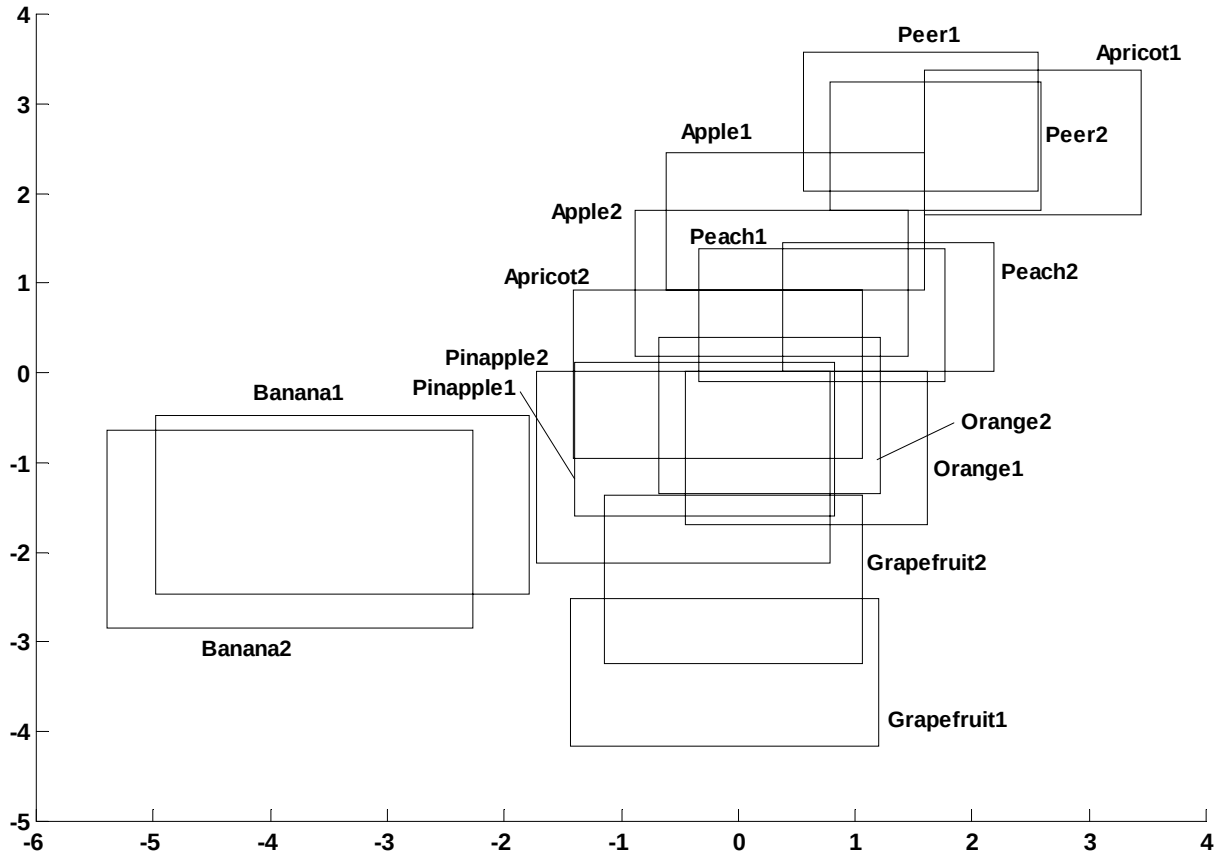
Figure 4: *Low dimensional representation of the fruit juices (first two components of C-PCA solution).*



The plots are consistent with the interpretation of the components. As one can observe from the figures, the positions of the fruit juices are rather similar. From the right, one can easily see the juices whose liking is the highest: the peer juices and Apricot 1. This group also has high sweetness and density values. This explains its position on the high right corner of the first principal plane. Two subgroups can be found below and on the left with respect to the above juices. Apple and peach juices form the first subgroup although Apple 1 liking is sensibly higher than those pertaining to the remaining juices. The latter subgroup is given by the fruits whose position is in the middle of Figures 4 and 5. Here, one can find the orange and pineapple juices and the remaining apricot juice (Apricot 2). Except for the latter one, the subgroup refers to the 'sour' fruits. Finally, the banana and grapefruit juices are positioned, respectively, on the left and below the latter group. All the ratings pertaining to the banana juices are low. Thus, the judges seem to dislike such juices, which have the lowest first component scores. Also the scores pertaining to the second component are quite low but, the grapefruit juices - displayed at the bottom of the plot – have the lowest second component scores. Their liking equals that of the 'sour' group but, as one may expect, the level of sweetness is the lowest. It is interesting to observe the existing hierarchy of the fruit juices along the second component. The sweetest fruits are on the top (except Apricot 2 which has particular features) and the sourest ones (orange and pineapple and, above all, grapefruit) on the

bottom.

Figure 5: *Low dimensional representation of the fruit juices (first two components of NN-PCA solution).*



Figures 4 and 5 also give a graphical representation of the widths of the original variables. Specifically, the size of the low dimensional hyperrectangles depends on the uncertainty associated to the most relevant variables with respect to the component at hand. The sizes of the hyperrectangles do not show sizeable differences in the low dimensional widths. However, we can approximately state that the more the liking of the fruit juices increases, the more the width regarding the first component decreases. Thus, when the judges like the juices, the ratings are systematically high; when the juices are less satisfactory, the judgements are fuzzier. On the contrary, such a relation does not hold with respect to the second component.

7. Conclusion

The present paper has reviewed three methods for recovering the underlying structure of fuzzy interval data: C-PCA, V-PCA and NN-PCA. In fact, C-PCA and V-PCA are suitable for interval valued data, but can be adopted for summarizing fuzzy intervals by ignoring the spread information. It has been explained how V-PCA can be made just as computationally efficient as C-PCA, by directly computing the cross-products matrix. We also described that, on the basis of the results, this approach does not modify the information needed for plotting the low dimensional hyperrectangles for the observation units: the minimum and maximum values of the vertex component

scores associated with each observation unit. NN-PCA exploits the tools of fuzzy arithmetics in order to compress fuzzy data sets. The approach is based upon a generalization for fuzzy data of the relationship between neural networks and PCA. The results of a simulation study were given in order to analyze the performance and the peculiarities of all the methods. It was found that it is impossible to state which is the best method in general. In fact, we observed that each method should be chosen according to the data structure (i.e., NN-PCA for simple loadings, C-PCA for nonsimple loadings), and the aims of the analysis. Finally, V-PCA, C-PCA and NN-PCA have been compared on the basis of an application to an empirical data set.
Future research can be considered in extending methods for interval valued and fuzzy data to deal with three-way data. In this respect, Giordani and Kiers (2004b) generalize C-PCA and V-PCA in a three-way context.

References

Baldi, P., and Hornik, K., (1989) Neural networks and principal component analysis: learning from examples without local minima, *Neural Networks*, **2**, 53-58.

Bock, H.H., and Diday, E., (2000) Analysis of symbolic data: exploratory methods for extracting statistical information from complex data. Springer Verlag, Heidelberg.

Boulard, H., and Kamp, Y., (1988) Auto-association by multilayer perceptrons and singular value decomposition, *Biol. Cybern.*, **59**, 291-294.

Cazes, P., (2002) Analyse factorielle d'un tableau de lois de probabilité. *Revue de Statistique Appliquée*, **50**, 5-24.

Cazes, P., Chouakria, A., Diday, E., and Schektman, Y., (1997) Extension de l'analyse en composantes principales à des données de type intervalle. *Revue de Statistique Appliquée*, **45**, 5-24.

Coppi, R., D'Urso, P., and Giordani, P. (2004), Component models for fuzzy data, Technical report n.1, Department of Statistics, Probability and Applied Statistics, University of Rome "La Sapienza". Submitted for publication.

Denœux, T., and Masson, M.H., (2004) Principal component analysis of fuzzy data using autoassociative neural networks, *IEEE Transactions on Fuzzy Systems*, **12**, 336-349.

D'Urso, P., and Giordani, P., (2004) A least squares approach to principal component analysis for interval valued data, *Chemometrics and Intelligent Laboratory Systems*, **70**, 179-192.

D'Urso, P., and Giordani, P., (2005) A possibilistic approach to latent component analysis for fuzzy data, *Fuzzy Sets and Systems*, **150**, 285-305.

Giordani, P., and Kiers, H.A.L., (2004a) Principal Component Analysis of symmetric fuzzy data, *Computational Statistics and Data Analysis*, **45**, 519-548.

Giordani, P., and Kiers, H.A.L., (2004b) Three-way component analysis of interval-valued data, *Journal of Chemometrics*, **18**, 253-264.

Kiers, H.A.L., (2000) Some procedures for displaying results from three-way methods, *Journal of Chemometrics*, **14**, 151-170.

Lauro C., and Palumbo F., (2000) Principal component analysis of interval data: a symbolic data analysis approach, *Computational statistics*, **15**, 73-87.

Palumbo, F., and Lauro, C., (2003) A PCA for interval-valued data based on midpoints and radii. In: Yanai H., Okada A., Shigemasu K., Kano Y., Meulman J., (Eds.), New Developments in Psychometrics (pp.641-648). Springer Verlag,

Tokyo.
Watada, J., and Yabuuchi, Y., (1997) Fuzzy principal component analysis and its application, *Biomed. Fuzzy Hum. Sci*., **3**, 83-92.
Zadeh, L.A., (1965) Fuzzy sets, *Informat*. *Control*, **8**, 338-353.
Zadeh, L.A., (1975) The concept of a linguistic variable and its application to approximate reasoning I, *Information Sciences*, **8**, 199-249.