



Nearest neighbour approach in the least-squares data imputation algorithms

I. Wasito, B. Mirkin *

*School of Computer Science and Information Systems, Birkbeck College, University of London,
Malet Street, London WC1E 7HX, UK*

Received 3 November 2002; received in revised form 18 December 2003; accepted 4 February 2004

Abstract

Imputation of missing data is of interest in many areas such as survey data editing, medical documentation maintaining and DNA microarray data analysis. This paper is devoted to experimental analysis of a set of imputation methods developed within the so-called least-squares approximation approach, a non-parametric computationally effective multidimensional technique. First, we review global methods for least-squares data imputation. Then we propose extensions of these algorithms based on the nearest neighbours approach. An experimental study of the algorithms on generated data sets is conducted. It appears that straight algorithms may work rather well on data of simple structure and/or with small number of missing entries. However, in more complex cases, the only winner within the least-squares approximation approach is a method, INI, proposed in this paper as a combination of global and local imputation algorithms.

© 2004 Elsevier Inc. All rights reserved.

Keywords: Least-squares; Nearest neighbour; Global–local learning; Singular value decomposition; Imputation

* Corresponding author. Tel.: +44-207-631-6746; fax: +44-207-631-6727.

E-mail addresses: itow@telkom.net (I. Wasito), mirkin@dcs.bbk.ac.uk, mirkin@dcs.bbk.ac.edu (B. Mirkin).

1. Introduction

The problem of imputation of missing data emerges in many areas such as survey data editing [5], medical documentation maintaining [17] and DNA microarray data modelling [31]. In the last decades a number of approaches have been proposed and utilised for filling in missing values. The most straightforward idea of using average values for imputation into missing entries, the Mean substitution [20], is probably the most popular approach. It has been supplemented recently with more refined versions such as hot/cold deck imputation and multidimensional techniques such as regression [19], decision trees [16,24], etc. Two other approaches, the maximum likelihood and least-squares approximation, take into account all available data to fill in all missings in parallel.

In the traditional statistics framework, any data set is considered as generated from a probabilistic distribution, which immediately leads to applying the maximum likelihood approach for modelling and imputation of incomplete data. This approach has led to introducing the so-called expectation–maximization (EM) method for handling incomplete data [6,28]. The EM algorithm provides a good framework both in theory and in practice. This approach, frequently supplemented with the so-called multiple imputation (MI) option, has proven to be a powerful tool [26–28]. However, the methods within this approach have two features that may become of issue in some situations. First, they may involve unsubstantiated hypotheses of the underlying distribution. Second, the rate of convergence of EM can be very slow sometimes. Furthermore, the computational cost of the method heavily depends on the absolute number of missing entries, and this can prevent its scalability to large databases.

Another multidimensional approach to imputation of missing data, the so-called least-squares approximation, extends the well-known matrix singular value decomposition (SVD) and, therefore, relies on the geometric structure of the data rather than on probabilistic properties. This approach is computationally effective and has attracted considerable attention of researchers [10,18,21]. However, this approach is not sensitive to the shape of the underlying distribution, which can become an issue in imputing missing data from a complex distribution.

In this paper, we experimentally explore computational properties of the least-squares approach and combine it with a machine learning approach, the so-called nearest neighbour (NN) method, which should balance the insensitivity of the least-squares mentioned above. Indeed, the NN approach suggests that, to impute a missing entry, only information of the nearest neighbours should be utilised, leaving other observations aside. This approach recently was successfully applied, in the context of bioinformatics, to the Mean substitution method in various settings [13,23,31]. We combine the local NN approach with

the least-squares algorithms to produce a number of NN versions of these algorithms. Further on, we develop a combined algorithm, INI, which applies an NN based least-squares technique to a preliminarily completed data matrix rather than to the original data with missings.

Then we experimentally explore the performances of eight least-squares imputation algorithms supplemented with Mean substitution method in both the global and local versions. In our experiments, we separately generate a data matrix and a missing pattern, which enables us to compare predicted values with those generated originally. It appears, performances of the algorithms depend on the type of data. In particular, when the underlying data model is simple enough, simple global least-squares imputation methods are best. However, in situations at which data structure is more complex, the local versions win. In particular, when the data structure is complex but proportion of missings is small, even the Mean substitution, in its local version, may be appropriate. However, with the number of missings growing, the method INI appears to be the only consistent winner.

The paper is organized as follows. Section 2 provides a review of existing techniques for handling missing data by categorising them in (a) prediction rules, (b) within the maximum likelihood framework and (c) the least-squares approximation. Section 3 gives a brief description of the global least-squares imputation methods. The NN versions of the imputation methods will be proposed in Section 4. The setting and results of our experimental study will be described in Section 5. Section 6 concludes the paper.

2. A review of imputation techniques

Among the methods of imputation of incomplete data, we can distinguish the following three approaches:

- (1) Prediction rules such as putting the variable mean into a missing entry or using regression or decision trees [19,20,24].
- (2) Maximum likelihood [6,20,27,28].
- (3) Least-squares approximation [10,12,18,21,32].

Let us discuss them in turn.

2.1. Prediction rules

The most popular method of imputation is substitution of a missing entry by the corresponding variable's mean, which will be referred to as Mean algorithm. An important drawback of the mean imputation is that the variance of the imputed data systematically underestimates the real variance [20]. The

Mean algorithm has been recently extended with other prediction models such as hot deck imputation at which the nearest neighbour's value is imputed, cold deck imputation at which the modal value is imputed, and regression imputation at which the regression-predicted value is imputed [19,20]. Decision trees are used for handling missing categorical data in [16,24]. Methods for imputation of incomplete data based on neural networks have been developed in [2,7]. Recently, the use of nearest-neighbour based techniques has been proposed for imputation of missing values [13,23,31].

The common feature of the prediction rule based approaches is that they rely on a limited number of variables. The other two approaches take advantage of using the entire available data entries to handle missings.

2.2. Maximum likelihood

The maximum likelihood approach relies on a parametric model of data generation, typically, multivariate Gaussian mixture model. A maximum likelihood method is applied for both fitting the model and imputation of the missing data. The most popular is the expectation–maximization (EM) algorithm [6,28]. It is sometimes implemented with multiple generation of candidates for missing entries according to the fitted probabilistic distribution, so that a missing entry is imputed as the candidates' mean. This is referred to as multiple imputation (MI) method [26,28]. EM/MI method have been implemented in many data imputation programs such as NORM and EMCOV which are freely available on the website [8]. The maximum likelihood approach is very popular since it is based on a precise statistical model. However, methods within this approach may involve unsubstantiated hypotheses and have a slow convergence rate. Either of these may prevent their scalability to large databases.

2.3. Least-squares approximation

This is a non-parametric approach based on approximation of the available data with a low-rank bilinear model akin to the singular value decomposition (SVD) of a data matrix.

Methods within this approach, typically, work sequentially by producing one factor at a time to minimize the sum of squared differences between the available data entries and those reconstructed via bilinear modelling. There are two ways to implement this approach:

- (1) Find an approximate data model using non-missing data only and then interpolate the missing values with values found with the model [10,12,21,29,32].

- (2) Start by filling in all the missing values, then iteratively approximate thus completed data and update the imputed values with those implied by the approximation [12,18].

3. Least-squares (LS) imputation techniques

In this section we describe generic methods within each of the two least-squares approximation approaches referred to above: (1) Iterative least-squares algorithm [10,12,21], (2) iterative majorization least-squares algorithm [12,18].

3.1. Notation

The data is considered in the format of a matrix \mathbf{X} with N rows and n columns. The rows are assumed to correspond to entities (observations) and columns to variables (features). The elements of a matrix \mathbf{X} are denoted by x_{ik} ($i = 1, \dots, N, k = 1, \dots, n$). The situation in which some entries (i, k) in \mathbf{X} may be missed is modelled with an additional matrix $\mathbf{M} = (m_{ik})$ where $m_{ik} = 0$ if the entry is missed and $m_{ik} = 1$, otherwise.

The matrices and vectors are denoted with boldface letters. A vector is always considered as a column; thus, the row vectors are denoted as transposes of the column vectors. Sometimes we show the operation of matrix multiplication with symbol $*$.

3.2. Iterative singular value decomposition

Let us describe the concept of singular value decomposition of a matrix (SVD) as a bilinear model for factor analysis of data. This model assumes the existence of a number $p \geq 1$ of hidden factors that underlie the observed data as follows:

$$x_{ik} = \sum_{t=1}^p c_{ik}z_{it} + e_{ik}, \quad i = 1, \dots, N, \quad k = 1, \dots, n \quad (1)$$

The vectors $\mathbf{z}_i = (z_{it})$ and $\mathbf{c}_i = (c_{ik})$ are referred to as factor scores for entities $i = 1, \dots, N$ and factor loadings for variables $k = 1, \dots, n$, respectively [15,21]. Values e_{ik} are residuals that are not explained by the model and should be made as small as possible.

To find approximating vectors $\mathbf{c}_i = (c_{ik})$ and $\mathbf{z}_i = (z_{it})$, we minimize the least-squares criterion:

$$L_2 = \sum_{i=1}^N \sum_{k=1}^n \left(x_{ik} - \sum_{t=1}^p c_{ik}z_{it} \right)^2 \quad (2)$$

It is proven that minimizing criterion (2) can be done with the following one-by-one strategy, which is, basically, the contents of the method of principal component analysis, one of the major data mining techniques [15].

According to this strategy, computations are carried out iteratively. At each iteration t , $t = 1, \dots, p$, only one factor is sought for. The criterion to be minimized at iteration t is:

$$l_2(\mathbf{c}, \mathbf{z}) = \sum_{i=1}^N \sum_{k=1}^n (x_{ik} - c_k z_i)^2 \quad (3)$$

with respect to the condition that the squared norm of \mathbf{c} is unity, that is, $\sum_{k=1}^n c_k^2 = 1$. It is well-known that solution to this problem is the singular triple $(\mu, \mathbf{z}, \mathbf{c})$ such that $\mathbf{X}\mathbf{c} = \mu\mathbf{z}$ and $\mathbf{X}^T\mathbf{z} = \mu\mathbf{c}$ with normed \mathbf{c} and μ equal to the norm of \mathbf{z} , so that $\mu = \sqrt{\sum_{i=1}^N z_i^2}$, the maximum singular value of \mathbf{X} . The found vectors \mathbf{c} and \mathbf{z} are stored as \mathbf{c}_t and \mathbf{z}_t , and next iteration $t + 1$ is performed. The matrix $\mathbf{X} = (x_{ik})$ changes from iteration t to iteration $t + 1$ by subtracting the found solution according to formula $x_{ik} \leftarrow x_{ik} - c_{ik} z_{it}$.

To minimize (3), the method of alternating minimization can be utilised. This method also works iteratively. Each iteration proceeds in two steps: (1) given a vector (c_k) , find optimal (z_i) ; (2) given (z_i) , find optimal (c_k) . Finding optimal score and loading vectors can be done according to equations:

$$z_i = \frac{\sum_{k=1}^n x_{ik} c_k}{\sum_{k=1}^n c_k^2} \quad (4)$$

and

$$c_k = \frac{\sum_{i=1}^N x_{ik} z_i}{\sum_{i=1}^N z_i^2} \quad (5)$$

that follow from the first-order optimality conditions.

This can be wrapped up as the following algorithm for finding a pre-specified number p of singular vectors.

Iterative SVD algorithm

0. Set number of factors p .
1. Set iteration number $t = 1$.
2. Initialize \mathbf{c}^* arbitrarily and normalize it. (Typically, we take $\mathbf{c}^{*t} = (1, \dots, 1)$.)
3. Given \mathbf{c}^* , calculate \mathbf{z} according to (4).
4. Given \mathbf{z} from step 3, calculate \mathbf{c} according to (5) and normalize it.
5. If $\|\mathbf{c} - \mathbf{c}^*\| < \epsilon$ where $\epsilon > 0$ is a pre-specified precision threshold go to 6; otherwise put $\mathbf{c}^* = \mathbf{c}$ and go to 3.
6. If $t < p$, set $t = t + 1$ and go to 7; otherwise End.
7. Set $\mu = \|\mathbf{z}\|$, $\mathbf{z}_t = \mathbf{z}/\mu$, and $\mathbf{c}_t = \mathbf{c}$; update $x_{ik} = x_{ik} - \mu c_{ik} z_{it}$ and go to step 2.

Note that \mathbf{z} is not normalised in the version of the algorithm described, which implies that its norm converges to the singular value μ , though the results are stored in the conventional normalized form. This method always converges if the initial \mathbf{c} does not belong to the subspace already taken into account in the previous singular vectors.

3.3. Iterative least-squares (ILS) algorithm for data imputation

The ILS algorithm is based on the SVD method described above. However, this time equation (1) applies only to those entries that are not missed.

The idea of the method is to find the score and loading vectors in decomposition (1) by using only those entries that are available and then use (1) for imputation of missing entries (with the residuals ignored).

To find approximating vectors $\mathbf{c}_t = (c_{ik})$ and $\mathbf{z}_t = (z_{it})$, we minimize the least-squares criterion on the available entries. The criterion can be written in the following form:

$$L_2 = \sum_{i=1}^N \sum_{k=1}^n e_{ik}^2 m_{ik} = \sum_{i=1}^N \sum_{k=1}^n \left(x_{ik} - \sum_{t=1}^p c_{tk} z_{it} \right)^2 m_{ik} \quad (6)$$

where $m_{ik} = 0$ at missings and $m_{ik} = 1$, otherwise.

To minimize criterion (6), the one-by-one strategy of the iterative SVD algorithm is utilised. According to this strategy, computations are carried out iteratively. At each iteration t , $t = 1, \dots, p$, only one factor is sought for to minimize criterion:

$$l_2 = \sum_{i=1}^N \sum_{k=1}^n (x_{ik} - c_k z_i)^2 m_{ik} \quad (7)$$

with respect to condition $\sum_{i=1}^N c_k^2 = 1$. The found vectors \mathbf{c} and \mathbf{z} are stored as \mathbf{c}_t and \mathbf{z}_t , non-missing data entries x_{ik} are substituted by $x_{ik} - c_k z_i$, and next iteration $t + 1$ is performed.

To minimize (7), the same method of alternating minimization is utilised. Each iteration proceeds in two steps: (1) given a vector (c_k) , find optimal (z_i) ; (2) given (z_i) , find optimal (c_k) . Finding optimal score and loading vectors can be done according to equations extending (4) and (5) to:

$$z_i = \frac{\sum_{k=1}^n x_{ik} m_{ik} c_k}{\sum_{k=1}^n c_k^2 m_{ik}} \quad (8)$$

and

$$c_k = \frac{\sum_{i=1}^N x_{ik} m_{ik} z_i}{\sum_{i=1}^N z_i^2 m_{ik}} \quad (9)$$

Basically, it is this procedure that was differently described in [10,12,21]. The following is a more formal presentation of the algorithm.

ILS algorithm

0. Set number of factors p .
1. Set iteration number $t = 1$.
2. Initialize n -dimensional $\mathbf{c}^{st} = (1, \dots, 1)$ and normalize it.
3. Given \mathbf{c}^* , calculate \mathbf{z} according to (8).
4. Given \mathbf{z} from step 3, calculate \mathbf{c} according to (9) and normalize it afterwards.
5. If $\|\mathbf{c} - \mathbf{c}^*\| < \epsilon$ where $\epsilon > 0$ is a pre-specified precision threshold go to 6; otherwise put $\mathbf{c}^* = \mathbf{c}$ and go to 3.
6. If $t < p$, store \mathbf{c} as \mathbf{c}_t and \mathbf{z} as \mathbf{z}_t , then set $t = t + 1$ and go to 7, otherwise go to 8.
7. For (i, k) such that $m_{ik} = 1$, update $x_{ik} = x_{ik} - c_{ik}z_{ik}$ and go to step 2.
8. Impute the missing values x_{ik} at $m_{ik} = 0$ according to (1) with $e_{ik} = 0$.

There are two issues which should be taken into account when implementing ILS:

(1) *Convergence*. The method may fail to converge depending on configuration of missings and starting point. Some other causes of non-convergence as those described by Grung and Manne [12] have been taken care of in our formulation of the algorithm. In the present approach we, somewhat simplistically, use the normed vector of ones as the starting point (step 2 in the algorithm above). However, sometimes a more sophisticated choice is required as the iterations may come to a “wrong convergence” or not converge at all. To this end, Gabriel and Zamir [10] developed a method to use a row of \mathbf{X} to build an initial \mathbf{c}^* , as follows:

1. Find (i, k) with the maximum

$$\omega_{ik} = \sum_b m_{bk}x_{bk}^2 + \sum_d m_{id}x_{id}^2 \quad (10)$$

over those (i, k) for which $m_{ik} = 0$.

2. With these i and k , compute

$$\beta = \frac{\sum_{b \neq i} \sum_{d \neq k} m_{bd}x_{bk}^2x_{id}^2}{\sum_{b \neq i} \sum_{d \neq k} m_{bd}x_{bk}x_{id}x_{bd}} \quad (11)$$

3. Set the following vector as initial at the ILS step 2:

$$\mathbf{c}^{st} = (x_{i1}, \dots, x_{ik-1}, \beta, x_{ik+1}, \dots, x_{in}) \quad (12)$$

The method is rather computationally intensive and may cause to slow down the speed of computation (up to 60 times in our experiments). However, it can be useful indeed when the size of the data is small.

(2) *Number of factors.* When the number of factors is equal to one, $p = 1$, ILS is equivalent to the method introduced by Wold [32] and his student Christoffersson [4] under the name of “nonlinear iterative partial least squares” (NIPALS). In most cases the one-factor imputation technique leads to significant errors, which implies the need for getting more factors. Selection of p may be driven by the same scoring function as selection of the number of principal components: the proportion of the data variance taken into account by the factors. This logic is well justified in the case of the principal component analysis at which model (1) fits the data exactly when p is equal to the rank of \mathbf{X} . When missings are present in the data, the number of factors sequentially found by ILS may be infinite; however, the logic is still justified since the residual data matrix converges to zero matrix (see Statement 2.2 in [21], which is applicable here).

3.4. Iterative majorization least-squares (IMLS) algorithm

This method is an example of application of the general idea that the weighted least-squares minimization problem can be addressed as a series of non-weighted least-squares minimization problems with iteratively adjusting solutions according to a so-called majorization function [14]. In this framework, Kiers [18] developed an algorithm, a version of which can be formulated within the framework presented. The algorithm starts with a completed data matrix and updates it by relying on both non-missing entries and estimates of missing entries.

The algorithm is similar to ILS except for the fact that it employs a different iterative procedure for finding a factor, that is, pair \mathbf{z} and \mathbf{c} , which will be referred to as Factor algorithm and described first. Factor algorithm operates with a completed version of matrix \mathbf{X} denoted by \mathbf{X}^s where $s = 0, 1, \dots$ is the iteration’s number. At each iteration s , the algorithm finds one best factor of SVD decomposition of \mathbf{X}^s and imputs the results into the missing entries, after which the next iteration starts.

Factor algorithm

1. Set $\mathbf{c}' = (1, \dots, 1)$ and normalize it.
2. Set $s = 0$ and define matrix \mathbf{X}^s by putting zeros into missing entries of \mathbf{X} . Set a measure of quality $h_s = \sum_{i=1}^N \sum_{k=1}^n x_{ik}^s$.
3. Find the first singular triple $\mathbf{z}_1, \mathbf{c}_1, \mu$ for matrix \mathbf{X}^s by applying the iterative SVD algorithm with $p = 1$ and denote the resulting value of criterion (6) by h_{s+1} . (Vectors $\mathbf{z}_1, \mathbf{c}_1$ are normalized here as described in the iterative SVD algorithm.)

4. If $|h_s - h_{s+1}| > \epsilon * h_s$ for a small $\epsilon > 0$, set $s = s + 1$, put $\mu z_{i1} c_{1k}$ for any missing entry (i, k) in \mathbf{X}^s and go back to step 3.
5. Set μz_1 and c_1 as the output.

Now we can formulate IMLS algorithm, which is a one-by-one version of the approach in [18], as follows.

IMLS algorithm

0. Set the number of factors p .
1. Set iteration number $t = 1$.
2. Apply Factor algorithm to matrix \mathbf{X} with the missing structure \mathbf{M} .
Denote results by z_t and c_t .
3. If $t = p$, go to step 5.
4. For (i, k) such that $m_{ik} = 1$, update $x_{ik} = x_{ik} - c_{ik} z_{it}$, put $t = t + 1$ and go to step 2.
5. Impute the missing values x_{ik} at $m_{ik} = 0$ according to (1) with $e_{ik} = 0$.

Theoretical properties of the IMLS method remain to be explored. IMLS has always converged in our experiments.

4. Nearest neighbour based data imputation

4.1. Lazy learning and nearest neighbour

The term “lazy learning” applies to a class of local learning techniques in which all the computation is performed in response to a request for prediction. The request is addressed by consulting data from only a relatively small number of entities considered relevant to the request according to a distance measure [1,3].

In this framework, the imputations are carried out sequentially, by analyzing entities with missing entries one-by-one. An entity containing one or more of missing entries which are to be imputed is referred to as a target entity. A most popular version of the lazy learning approach is the so-called K -nearest neighbour (see, for instance, [22]). According to this approach, a distance measure is computed between the target entity and each of the other entities and then K entities nearest to the target are selected. The imputation model such as (1), for the target entity, is found by using a shortened version of \mathbf{X} to contain only $K + 1$ elements: the target and K selected neighbours.

To apply the NN approach, the following two issues should be addressed.

- (1) *Measuring distance*. There can be a multitude of distance measures considered. We choose Euclidean distance squared as this measure is compatible

with the least-squares framework. The distance between a target entity \mathbf{X}_i and an entity \mathbf{X}_j is defined as:

$$D_2(\mathbf{X}_i, \mathbf{X}_j, \mathbf{M}) = \sum_{k=1}^n [x_{ik} - x_{jk}]^2 m_{ik} m_{jk}; \quad i, j = 1, 2, \dots, N \quad (13)$$

where m_{ik} and m_{jk} are missingness values for x_{ik} and x_{jk} , respectively. This distance was also used in [13,23,31].

- (2) *Selection of the neighbourhood.* The principle of selecting the closest entities can be realized, first, as is, on the set of all entities, and, second, by considering only entities with non-missing entries in the attribute corresponding to that of the target's missing entry. The second approach was applied in [13,31] for data imputation with the method Mean. We apply the same approach when using this method. However, for ILS and IMLS, the presence of missing entries in the neighbours creates no problems, and, with these methods, we select among all entities.

Further we consider nearest neighbour (NN) based versions for the algorithms above.

4.2. Nearest neighbour imputation algorithms

Briefly, the NN based techniques can be formulated as follows: take the first row that contains a missing entry as the target entity \mathbf{X}_i , find its K nearest neighbours, and form a matrix \mathbf{X} consisting of the target entity and the neighbours. Then apply an imputation algorithm to the matrix \mathbf{X} with imputing missing entries at the target entity only. Repeat this until all missing entries are filled in. Then output the completed data matrix.

NN based imputation algorithm

1. Take the first row in the data table that contains a missing entry as the target entity \mathbf{X}_i .
2. Find K neighbours of \mathbf{X}_i .
3. Create a data matrix \mathbf{X} consisting of \mathbf{X}_i and K selected neighbours.
4. Apply an imputation algorithm to matrix \mathbf{X} and impute missing values in \mathbf{X}_i .
5. If there are no missing entries in the data, then stop; otherwise go back to step 1.

To make the NN based imputation algorithms work fast, we choose K be of the order of 5–10 entities. Then, to apply the least-squares imputation techniques, we have to restrict ourselves to small numbers of factors to guarantee that the subspace approximation processes converge. Thus, we take $p = 1$ and

use the Gabriel–Zamir’s initialization in ILS. Still, ILS algorithm may lead to non-convergent results because of the small NN data sizes. In contrast, IMLS algorithm always uses complete data and converges.

4.3. Global–local learning imputation algorithm

One more NN based approach can be suggested to combine the lazy learning approach with the global imputation algorithms described in Section 3. In this approach, we use a global imputation technique to fill in all the missings in matrix \mathbf{X} . Let us denote the resulting matrix \mathbf{X}^* . Then a lazy learning technique is applied to fill in the missings in \mathbf{X} again, but, this time, based on distances computed with the completed data \mathbf{X}^* .

The same distance formula (13) can be utilised in this case as well, by assuming that all values m_{ik} are unities, which is the case when matrix \mathbf{X}^* is utilised. This distance will be referred to as the *prime distance*.

Let us consider an application of this global–local approach involving IMLS at both of the stages. This technique, referred to as INI in the remainder, will include four main steps. Firstly, impute missing values in the data matrix \mathbf{X} by using IMLS with $p = 4$. Then compute the prime distance metric with thus found \mathbf{X}^* . Take a target entity according to \mathbf{X} and apply the NN algorithm to find its neighbours according to \mathbf{X}^* . Finally, impute all the missing values in the target entity with NN based IMLS technique (this time, with $p = 1$).

INI algorithm

1. Apply IMLS algorithm to \mathbf{X} with $p = 4$ to impute all missing entries in matrix \mathbf{X} ; denote resulting matrix by \mathbf{X}^* .
2. Take the first row in \mathbf{X} that contains a missing entry as the target entity \mathbf{X}_i .
3. Find K neighbours of \mathbf{X}_i on matrix \mathbf{X}^* .
4. Create a data matrix \mathbf{X}_c consisting of \mathbf{X}_i and rows of \mathbf{X} corresponding to the selected K neighbours.
5. Apply IMLS algorithm with $p = 1$ to \mathbf{X}_c and impute missing values in \mathbf{X}_i of \mathbf{X} .
6. If no missing entries remain, stop; otherwise go back to step 2.

5. Experimental study

The goal of the experimental study is twofold:

- (1) To compare different methods of imputation on various data sets and missing patterns.
- (2) To see whether performances of NN based techniques are always superior to those of global least-squares techniques.

5.1. Selection of algorithms

The considerations above lead us to consider the following eight least-squares data imputation algorithms as a representative selection:

- (1) ILS-NIPALS or NIPALS: ILS with $p = 1$.
- (2) ILS: ILS with $p = 4$.
- (3) ILS-GZ or GZ: ILS with the Gabriel–Zamir procedure for initial settings.
- (4) IMLS-1: IMLS with $p = 1$.
- (5) IMLS-4: IMLS with $p = 4$.
- (6) N-ILS: NN based ILS with $p = 1$.
- (7) N-IMLS: NN based IMLS-1.
- (8) INI: NN based IMLS-1 imputation based on distances from an IMLS-4 imputation.

Of these, the first five are versions of the global ILS and IMLS methods, the next two are lazy learning versions of the same approaches, and the last algorithm INI combines local and global versions of IMLS. Similar combined algorithms involving ILS have been omitted here since they do not always converge. For the purposes of comparison, two mean scoring algorithms have been added:

- (9) Mean: Imputing the average column value.
- (10) N-Mean: NN based Mean.

In the follow-up experiments, the NN based techniques will operate with $K = 10$.

5.2. Data generation

5.2.1. Rank one data model

Let us specify some vectors $\mathbf{c}_{(15 \times 1)}$, $\mathbf{z}_{(200 \times 1)}$ with their components in the range between -1 and 1 , and generate a uniformly random matrix $\mathbf{E}_{(200 \times 15)}$ within the same range. Then the data model can be formulated as follows:

$$\mathbf{X}_\epsilon = \mathbf{z} * \mathbf{c}' + \epsilon \mathbf{E} \quad (14)$$

where coefficient ϵ scales the random noise added to the one-dimensional matrix $\mathbf{z} * \mathbf{c}'$. The coefficient ϵ will be referred to as the noise level; it has been taken at six levels from $\epsilon = 0.1$ to $\epsilon = 0.6$.

Model (14) can be applied as many times as a data set is needed.

5.2.2. Gaussian mixture data model

Gaussian mixture data model is described in many monographs (see, for instance, [9]). In this model, a data matrix $\mathbf{D}_{N \times n}$ is generated randomly from the Gaussian mixture distribution with a probabilistic principal component analysis (PCA) covariance matrix [25,30]. We refer to a mixture of p Gaussian distributions (classes) as a Gaussian p -mixture. The following three-step procedure, Neural Network NETLAB, is applied as implemented in a MATLAB Toolbox freely available on the web [11]:

- (1) Architecture: set the dimension of data equal to n , number of classes (Gaussian distributions) to p and the type of covariance matrix based on the probabilistic PCA in a q dimension subspace. In our experiments, p is 3 or 5, n between 15 and 25, and q typically is $n - 3$.
- (2) Data structure: create a Gaussian mixture model with the mixing coefficient equal to $1/p$ for each class. A Gaussian distribution for each i th class ($i = 1, \dots, p$) is defined as follows: a random n -dimensional vector \mathbf{avg}_i is generated based on Gaussian distribution $N(0, 1)$. The $n \times q$ matrix of the first q loading n -dimensional vectors is defined as follows:

$$\mathbf{W}_q = \begin{pmatrix} \mathbf{I}_{q \times q} \\ \mathbf{1}_{(n-q) \times q} \end{pmatrix} \quad (15)$$

where $\mathbf{I}_{q \times q}$ and $\mathbf{1}_{(n-q) \times q}$ are the identity matrix and matrix of ones, respectively.

In our experiments, the general variance σ^2 is set to be equal to 0.1. The probabilistic PCA (PPCA) covariance matrix is computed as follows:

$$\mathbf{Cov} = \mathbf{W}_q * \mathbf{W}_q' + \sigma^2 \mathbf{I}_{n \times n} \quad (16)$$

- (3) Data: generate randomly data matrix $\mathbf{D}_{N \times n}$ from the Gaussian mixture distribution defined above, as follows:

Compute eigenvectors and corresponding eigenvalues of \mathbf{Cov} and denote the matrix of eigenvectors by \mathbf{evec} and vector of the square roots of eigenvalues by $\sqrt{\mathbf{eigen}}$.

For $i = 1, \dots, p$:

Set $N_i = N/p$, the number of rows in i th class.

Generate randomly $\mathbf{R}_{(N_i \times n)}$ based on Gaussian distribution $N(0, 1)$.

Compute $\mathbf{D}_i = \mathbf{avg}_i + \mathbf{R} * \mathbf{diag}(\sqrt{\mathbf{eigen}}) * \mathbf{evec}'$.

end

Define \mathbf{D} as $N \times n$ matrix combining all generated matrices \mathbf{D}_i , $i = 1, \dots, p$.

5.2.3. Generation of missings

Given a data table generated, a pattern of missing entries is produced randomly on a matrix of the size of the data table with a pre-specified proportion of the missings. The proportion of missing entries may vary. We use the random uniform distribution for generating missing positions and the proportion's range at 1%, 5%, 10%, 15%, 20% and 25% of the total number of entries.

5.2.4. Evaluation of results

Since the data and missings are generated separately, we can evaluate the quality of imputation by comparing the imputed values with those generated at the stage of data generating. We use the squared imputation error, IE, to measure the performance of an algorithm. The measure is defined as follows:

$$IE = \frac{\sum_{i=1}^N \sum_{k=1}^n (1 - m_{ik})(x_{ik} - x_{ik}^*)^2}{\sum_{i=1}^N \sum_{k=1}^n (1 - m_{ik})x_{ik}^2} \quad (17)$$

where m_{ik} is the missingness matrix entry and x_{ik}^* the data matrix \mathbf{X}^* with imputed values.

5.3. Experimental results

5.3.1. Experiments with rank one data model

Table 1 shows the average results of 30 experiments (five data sets times six missings patterns) with the selected algorithms. Both ILS and ILS-GZ frequently do not converge, probably because of too many factors, four, required in them. This is labelled by the symbol 'N/A' put in the corresponding cells of the Table 1. Table 1 shows that, in each of the methods, except the Mean, the error increases with the noise level growing. At the Mean, the error stands constant on the level of about 100%. Two obvious winners, according to the table, are NIPALS (that is, ILS-1) and IMLS-1. The other methods' performances are much worse. This, probably, can be explained by the very nature of the data generated: unidimensionality. The other methods seem just over-specified and thus wrong in this situation.

To remove the effect of failing convergencies and, moreover, the effect of overlapping dispersions in performances of the methods, we also compare the methods pairwise. That is, for each pair of methods we consider at how many experiments one of them outperformed the other. These results are shown in Table 2 (i, j) entry in which shows how many times, per cent, the method in j th column outperformed the method in i th row.

The data in Table 2 confirm the results in Table 1. Moreover, we see that IMLS-1 was the winner more often than NIPALS (76% to 24%). Also, method

Table 1

The average squared errors of imputation (%) for different methods (in rows) at different noise levels, columns; the values in parentheses are corresponding standard deviations, per cent as well

Methods	Noise level					
	0.1	0.2	0.3	0.4	0.5	0.6
ILS	33.45 (179.42)	110.38 (478.79)	235.06 (928.57)	N/A (N/A)	N/A (N/A)	N/A (N/A)
GZ	N/A (N/A)	N/A (N/A)	282.97 (1042)	359.19 (1416)	303.12 (1413)	260.17 (1180)
NIPALS	3.44 (0.58)	12.67 (2.17)	25.09 (4.44)	38.12 (6.98)	50.12 (9.33)	60.41 (11.14)
IMLS-1	3.44 (0.58)	12.67 (2.17)	25.07 (4.44)	38.09 (6.98)	50.08 (9.32)	60.35 (11.13)
IMLS-4	34.37 (82.54)	17.69 (3.02)	34.92 (6.26)	53.18 (9.76)	69.70 (12.84)	84.10 (15.40)
Mean	100.77 (1.21)	100.73 (1.21)	100.69 (1.23)	100.65 (1.25)	100.61 (1.29)	100.59 (1.32)
N-ILS	N/A (N/A)	N/A (N/A)	N/A (N/A)	N/A (N/A)	N/A (N/A)	N/A (N/A)
N-IMLS	13.38 (23.20)	20.61 (13.21)	43.76 (36.70)	61.29 (31.05)	84.89 (36.84)	103.56 (43.02)
INI	33.84 (83.45)	28.08 (42.19)	42.26 (55.89)	53.55 (34.84)	63.86 (16.39)	78.75 (21.27)
N-Mean	74.30 (45.43)	77.55 (41.32)	82.08 (35.89)	87.53 (30.95)	92.57 (27.36)	97.83 (24.53)

Table 2

The pair-wise comparison between 10 methods; an entry (i, j) shows how many times, per cent, method j outperformed method i with the rank one data model

	ILS	GZ	NIPALS	IMLS-1	IMLS-4	Mean	N-ILS	N-IMLS	INI	N-Mean
ILS	–	44.00	100.00	100.00	74.00	1.50	32.00	80.00	98.00	20.00
GZ	56.00	–	100.00	100.00	77.00	1.50	33.00	88.00	98.00	17.00
NIPALS	0.00	0.00	–	76.00	0.00	0.00	0.00	0.00	0.00	0.00
IMLS-1	0.00	0.00	24.00	–	0.00	0.00	0.00	0.00	0.00	0.00
IMLS-4	26.00	23.00	100.00	100.00	–	0.00	21.00	67.00	97.50	0.00
Mean	98.50	98.50	100.00	100.00	100.00	–	52.00	99.50	100.00	77.00
N-ILS	68.00	67.00	100.00	100.00	79.00	48.00	–	94.00	100.00	67.00
N-IMLS	20.00	12.00	100.00	100.00	33.00	0.50	6.00	–	92.00	3.00
INI	2.00	2.00	100.00	100.00	2.50	0.00	0.00	8.00	–	0.00
N-Mean	80.00	83.00	100.00	100.00	100.00	23.00	33.00	97.00	100.00	–

INI gets noted as the third ranking winner, which should be attributed to the fact that it heavily relies on IMLS-1.

5.3.2. Experiments with Gaussian mixture data model

The experiments are carried out with two types of generated data: 3-mixture and 5-mixture, both with the dimension of the PPCA subspace equal to $n - 3$. We generated 10 data sets of random sizes (from 200 to 250 rows and from 15 to 25 columns) for each of these two data types. We also generated six sets of missing patterns for each of the following levels of data missing: 1%, 5%, 10%, 15%, 20% and 25% missings. Altogether, we have thus 36 various patterns of missings. The results of running all the 10 imputation methods over each of the 10 data sets with the 36 missing patterns will be presented in the same format as in the previous section.

5.3.2.1. Results for 3-mixture data. Table 3 shows the average results of 60 experiments (10 data set times six missing patterns) in analysis of the selected algorithms. The results for N-ILS reflect somewhat poor convergence of the method at the levels of missing greater than 1%. Once again we see that, for each of the algorithms, except the Mean, the error increases as the number of missings grows. However, the increase is at a smaller scale than at the rank one data in the previous section. At the Mean method, the error is constant again, of the order of 92%, also a decrease from the results for the rank one data.

The obvious winner, at each level of missings, is INI, the global–local version of least-squares imputation. In a close range, it is followed by IMLS-4 and ILS-GZ. Method N-IMLS is around: it is the second best when missings are sparse, but it falls out of the range when the proportion of missings grows to

Table 3

The average squared error of imputation at 3-mixture data with different levels of missing entries (%)

Methods	Proportion of missing					
	1%	5%	10%	15%	20%	25%
ILS	31.45 (15.99)	29.52 (7.28)	30.69 (4.68)	33.27 (7.33)	36.98 (8.35)	40.03 (8.49)
GZ	31.45 (15.99)	29.52 (7.28)	30.67 (4.66)	33.26 (7.34)	37.00 (7.34)	40.09 (8.47)
NIPALS	49.86 (17.71)	50.08 (10.55)	50.11 (6.28)	52.33 (8.03)	54.41 (9.53)	56.03 (8.97)
IMLS-1	49.86 (17.73)	49.70 (10.01)	49.75 (6.21)	50.45 (5.33)	50.63 (4.98)	51.42 (4.54)
IMLS-4	31.45 (16.06)	29.13 (6.11)	30.25 (4.54)	31.30 (3.65)	33.16 (4.14)	35.30 (3.89)
Mean	93.41 (12.06)	91.76 (3.77)	92.58 (3.66)	92.15 (2.88)	92.30 (2.63)	92.26 (2.13)
N-ILS	30.00 (100.00)	300.00 (2100)	880.00 (4000)	N/A (N/A)	N/A (N/A)	N/A (N/A)
N-IMLS	30.63 (15.36)	29.09 (7.16)	30.38 (4.51)	35.84 (9.74)	44.83 (16.33)	54.99 (18.78)
INI	29.96 (15.07)	28.18 (7.21)	28.51 (4.37)	29.96 (3.93)	33.18 (5.82)	35.10 (5.58)
N-Mean	33.13 (12.98)	39.94 (8.42)	52.88 (6.55)	63.75 (8.92)	74.65 (9.50)	86.04 (11.68)

25%. These conclusions may be blurred by the overlapping standard deviations of the methods' average errors. Therefore, once again, let us consider results of direct pairwise comparisons between the methods. At this perspective, the results appear to depend on the level of missings: there are somewhat different situations at 1% missings and at the other, greater, missing levels, which are similar to each other. Therefore, Table 4 will present results of the pair-wise comparisons at 1% missings and Table 5 will show a typical pattern of pair-wise comparisons between the methods at a greater level of missings (15%, in this case).

Table 4 shows that with 1% missings, all the three nearest neighbour versions of the least-squares imputation, N-ILS, N-IMLS, and INI, have very good and similar performances, though N-ILS is obviously weaker than N-IMLS (by 10–90%). This shows that N-ILS, actually, is not that bad as it is seen in Table 3: it just does not always converge, but when it converges, it works well.

Table 4

The pair-wise comparisons of methods; an entry (i, j) shows how many times in % method j performed better than method i on a mixture of three with $n - 3$ PPCA factors Gaussian distributions for 1% missing data

	ILS	GZ	NIPALS	IMLS-1	IMLS-4	Mean	N-ILS	N-IMLS	INI	N-Mean
ILS	–	30.00	10.00	10.00	60.00	10.00	60.00	60.00	70.00	40.00
GZ	70.00	–	10.00	10.00	60.00	10.00	60.00	60.00	70.00	40.00
NIPALS	90.00	90.00	–	50.00	90.00	0.00	90.00	90.00	90.00	80.00
IMLS-1	90.00	90.00	50.00	–	90.00	0.00	90.00	90.00	90.00	80.00
IMLS-4	40.00	40.00	10.00	10.00	–	10.00	60.00	60.00	70.00	40.00
Mean	90.00	90.00	100.00	100.00	90.00	–	100.00	100.00	90.00	100.00
N-ILS	40.00	40.00	10.00	10.00	40.00	0.00	–	90.00	50.00	40.00
N-IMLS	40.00	40.00	10.00	10.00	40.00	0.00	10.00	–	50.00	40.00
INI	30.00	30.00	10.00	10.00	30.00	10.00	50.00	50.00	–	40.00
N-Mean	60.00	60.00	20.00	20.00	60.00	0.00	60.00	60.00	60.00	–

Table 5

The pair-wise comparisons of methods; an entry (i, j) shows how many times in % method j performed better than method i on a mixture of three with $n - 3$ PPCA factors Gaussian distributions for 15% missing data

	ILS	GZ	NIPALS	IMLS-1	IMLS-4	Mean	N-ILS	N-IMLS	INI	N-Mean
ILS	–	20.00	0.00	0.00	70.00	0.00	20.00	20.00	80.00	0.00
GZ	80.00	–	0.00	0.00	70.00	0.00	20.00	20.00	80.00	0.00
NIPALS	100.00	100.00	–	70.00	100.00	0.00	50.00	100.00	100.00	0.00
IMLS-1	100.00	100.00	30.00	–	100.00	0.00	50.00	100.00	100.00	0.00
IMLS-4	30.00	30.00	0.00	0.00	–	0.00	20.00	20.00	70.00	0.00
Mean	100.00	100.00	100.00	100.00	100.00	–	70.00	100.00	100.00	100.00
N-ILS	80.00	80.00	50.00	50.00	80.00	30.00	–	90.00	100.00	30.00
N-IMLS	80.00	80.00	0.00	0.00	80.00	0.00	10.00	–	100.00	0.00
INI	20.00	20.00	0.00	0.00	30.00	0.00	0.00	0.00	–	0.00
N-Mean	100.00	100.00	100.00	100.00	100.00	0.00	70.00	100.00	100.00	–

This mixed result does not hold at greater levels of missings. Table 5 shows that INI becomes the single winner, with no close runner-ups. Also, Mean consistently displays the worst performance. Overall, results reported in Table 5 are consistent with those in Table 4.

5.3.2.2. Results for 5-mixture data. Let us consider results of the experiments at data generated from Gaussian 5-mixture distributions. The average error's pattern is presented in Table 6. Obviously, the pattern is almost identical to that of results at the Gaussian 3-mixture data in Table 4. This implies similar conclusions to those reported above according to Table 4.

However, this flat pattern becomes more complex when we analyse results of pairwise comparisons between the methods at different levels of missings. We can see three different patterns in pairwise comparisons: (1) at 1% missings (Table 7), (2) at 5% missings (Table 8), and (3) at 10% and more missings (Table 9).

At 1% missings, according to Table 7, there are four winners, all the nearest neighbour based methods, N-Mean included. Although N-Mean loses to INI by 30–70%, it outperforms the others in winning over one-dimensional NIPALS and IMLS-1. This shows that the local version of Mean method can be rather appropriate when the proportion of missings is small.

Unfortunately, when the proportion of missings increases to 5% and more, the N-Mean method loses to all the least-squares imputation methods except for the unidimensional ones, NIPALS and IMLS-1.

Table 8 shows results of pairwise comparison of the methods at 5% missings. This time, there are only three winners, INI, N-IMLS and N-ILS, that are ordered in such a way that the previous one wins over the next one(s) in 70% of

Table 6

The average squared error of imputation and its standard deviation (%) at 5-mixture data with different levels of missing entries

Methods	Proportion of missings					
	1%	5%	10%	15%	20%	25%
ILS	41.25 (15.00)	39.66 (7.91)	38.68 (4.64)	40.42 (6.00)	45.33 (7.82)	48.56 (9.58)
GZ	41.26 (15.01)	39.66 (7.91)	38.67 (4.67)	40.43 (6.02)	45.27 (7.83)	48.25 (8.99)
NIPALS	56.49 (20.12)	50.64 (10.27)	49.54 (5.46)	49.75 (6.76)	54.14 (8.88)	55.25 (9.51)
IMLS-1	56.59 (20.19)	50.59 (10.35)	49.38 (5.52)	48.93 (4.82)	51.75 (5.49)	52.35 (5.80)
IMLS-4	41.25 (14.99)	39.66 (8.11)	38.68 (4.56)	40.42 (4.37)	45.33 (4.50)	48.56 (6.41)
Mean	97.13 (8.50)	95.82 (2.94)	96.19 (3.01)	96.11 (1.89)	95.72 (1.90)	95.62 (1.51)
N-ILS	35.14 (1000)	193.02 (12200)	N/A (N/A)	N/A (N/A)	N/A (N/A)	N/A (N/A)
N-IMLS	35.04 (13.96)	34.71 (7.83)	36.80 (7.64)	41.82 (8.67)	53.58 (17.38)	66.75 (19.06)
INI	35.29 (13.03)	33.19 (6.81)	33.27 (4.83)	34.89 (4.75)	38.93 (5.47)	43.01 (8.05)
N-Mean	37.66 (13.19)	41.98 (7.32)	50.96 (5.67)	59.32 (5.91)	69.70 (7.80)	80.98 (8.58)

Table 7

The pair-wise comparisons of methods; an entry (i, j) shows how many times in % method j outperformed method i on Gaussian 5-mixtures with $n - 3$ PPCA factors for 1% missing data

	ILS	GZ	NIPALS	IMLS-1	IMLS-4	Mean	N-ILS	N-IMLS	INI	N-Mean
ILS	–	70.00	0.00	0.00	30.00	0.00	60.00	60.00	60.00	60.00
GZ	30.00	–	0.00	0.00	30.00	0.00	60.00	60.00	60.00	60.00
NIPALS	100.00	100.00	–	20.00	90.00	10.00	90.00	90.00	100.00	100.00
IMLS-1	100.00	100.00	80.00	–	90.00	10.00	90.00	90.00	100.00	100.00
IMLS-4	70.00	70.00	10.00	10.00	–	0.00	70.00	70.00	80.00	60.00
Mean	100.00	100.00	90.00	90.00	100.00	–	100.00	100.00	100.00	100.00
N-ILS	40.00	40.00	10.00	10.00	30.00	0.00	–	100.00	50.00	50.00
N-IMLS	40.00	40.00	10.00	10.00	30.00	0.00	0.00	–	50.00	50.00
INI	40.00	40.00	0.00	0.00	20.00	0.00	50.00	50.00	–	30.00
N-Mean	40.00	40.00	0.00	0.00	40.00	0.00	50.00	50.00	70.00	–

Table 8

The pair-wise comparisons of methods; an entry (i, j) shows how many times in % method j outperformed method i on Gaussian 5-mixtures with $n - 3$ PPCA factors for 5% missing data

	ILS	GZ	NIPALS	IMLS-1	IMLS-4	Mean	N-ILS	N-IMLS	INI	N-Mean
ILS	–	40.00	0.00	0.00	40.00	0.00	80.00	80.00	100.00	10.00
GZ	60.00	–	0.00	0.00	40.00	0.00	80.00	80.00	100.00	10.00
NIPALS	100.00	100.00	–	50.00	100.00	0.00	100.00	100.00	100.00	90.00
IMLS-1	100.00	100.00	50.00	–	100.00	0.00	100.00	100.00	100.00	90.00
IMLS-4	60.00	60.00	0.00	0.00	–	0.00	80.00	90.00	100.00	20.00
Mean	100.00	100.00	100.00	100.00	100.00	–	100.00	100.00	100.00	100.00
N-ILS	20.00	20.00	0.00	0.00	20.00	0.00	–	70.00	70.00	20.00
N-IMLS	20.00	20.00	0.00	0.00	10.00	0.00	30.00	–	70.00	20.00
INI	0.00	0.00	0.00	0.00	0.00	0.00	30.00	30.00	–	0.00
N-Mean	90.00	90.00	10.00	10.00	80.00	0.00	80.00	80.00	100.00	–

Table 9

The pair-wise comparisons of methods; an entry (i, j) shows how many times in % method j outperformed method i on Gaussian 5-mixtures with $n - 3$ PPCA factors for 15% missing data

	ILS	GZ	NIPALS	IMLS-1	IMLS-4	Mean	N-ILS	N-IMLS	INI	N-Mean
ILS	–	40.00	0.00	0.00	50.00	0.00	40.00	50.00	100.00	0.00
GZ	60.00	–	0.00	0.00	50.00	0.00	40.00	60.00	100.00	0.00
NIPALS	100.00	100.00	–	30.00	100.00	0.00	60.00	100.00	100.00	10.00
IMLS-1	100.00	100.00	70.00	–	100.00	0.00	70.00	100.00	100.00	10.00
IMLS-4	50.00	50.00	0.00	0.00	–	0.00	40.00	60.00	90.00	0.00
Mean	100.00	100.00	100.00	100.00	100.00	–	80.00	100.00	100.00	100.00
N-ILS	60.00	60.00	40.00	30.00	60.00	20.00	–	100.00	100.00	20.00
N-IMLS	50.00	40.00	0.00	0.00	40.00	0.00	0.00	–	100.00	0.00
INI	0.00	0.00	0.00	0.00	10.00	0.00	0.00	0.00	–	0.00
N-Mean	100.00	100.00	90.00	90.00	100.00	0.00	80.00	100.00	100.00	–

the cases. Thus, INI leads the contest and Mean loses it on almost every count, at the 5% proportion of missings.

When the proportion of missings grows further on, INI becomes the only winner again, as can be seen from Table 9 presenting a typical pattern. Another feature of the pattern is that ILS, GZ and IMLS-4 perform similarly to the local versions, N-ILS and N-IMLS. Once again, the method Mean is the worst, this time always losing to all other methods.

5.4. Performance

As we mentioned already, the data table sizes in our experiments have been about $(200\text{--}250) \times (15\text{--}25)$. With this type of data, the ordinary global least-squares methods such as ILS require about half a second to run on the platform MatLab-6 installed at a Pentium III 733 MHz. The nearest neighbour versions of least-squares imputation are about 30 times slower than that. ILS-GZ appears to be the slowest of the 10 methods.

5.5. Data complexity

As was shown above, the methods may perform differently depending on data complexity. For instance, the ordinary least-squares imputation methods perform better than the local versions when the data is generated from a unidimensional source.

Thus, a measure of data complexity should be introduced to give the user a guidance in prior selection of appropriate imputation methods.

A good candidate for such a measure seems to be the contribution of the first factor to the data scatter. As is well known, a singular value squared shows the part of the total data scatter taken into account by the corresponding principal component. The relative contribution of h th factor to the data scatter, thus, is equal to

$$\text{Contribution}_h = \frac{\mu_h^2}{\sum_{i=1}^N \sum_{k=1}^n x_{ik}^2} \quad (18)$$

where μ_h is h th singular value of matrix \mathbf{X} . This measure can be extended to the case of missing data, as well.

The proportion of the first greatest component shows how much the rank of the data matrix is close to 1: the larger Contribution_1 the closer. The data matrix is simplest when it has rank 1, that is, $\text{Contribution}_1 = 1$.

Especially appealing this measure seems as a device to distinguish between the one-dimensional and Gaussian data generators. However, the measure does not necessarily work. To demonstrate this, let us compare contributions of the first factor to data generated according to the one-dimensional model

Table 10

Contribution of the first factor to the data scatter (%) for the unidimensional data generator

Data sets	Noise level					
	0.1	0.2	0.3	0.4	0.5	0.6
Data-1	97.39	90.28	80.54	70.11	60.31	51.78
Data-2	96.95	88.84	78.10	67.03	56.99	48.48
Data-3	96.92	88.89	78.45	67.77	58.10	49.89
Data-4	97.61	91.14	82.18	72.42	63.05	54.69
Data-5	97.15	89.61	79.57	69.09	59.41	51.06

Table 11

Contributions of the first two factors to the data scatter, %, for the Gaussian 3-mixture data

Data sets	Contribution (%)		
	First	Second	Total
Data-1	59.33	12.49	71.82
Data-2	59.27	12.27	71.54
Data-3	52.58	17.03	69.61
Data-4	54.80	14.01	68.81
Data-5	58.64	11.77	70.41
Data-6	58.42	13.53	71.95
Data-7	59.17	11.90	71.17
Data-8	56.85	13.21	70.06
Data-9	59.05	11.59	70.64
Data-10	57.22	10.23	67.45

Table 12

Contributions of the first two factors to the data scatter, %, for the Gaussian 5-mixture data

Data sets	Contribution (%)		
	First	Second	Total
Data-1	57.93	8.26	66.19
Data-2	62.73	8.37	71.10
Data-3	60.91	7.15	68.06
Data-4	58.88	7.81	66.69
Data-5	58.41	10.81	69.22
Data-6	61.50	8.79	70.29
Data-7	59.69	11.27	70.96
Data-8	53.64	9.79	63.43
Data-9	58.33	13.76	72.09
Data-10	58.42	9.60	68.02

(see Table 10) and to data generated according to Gaussian 3- and 5-mixture models (Tables 11 and 12).

Table 10 presents contributions of the first factor to data sets generated according to the unidimensional model at the noise levels from $\epsilon = 0.1$ through

$\epsilon = 0.6$. The contribution falls from 97% to 52%, on average, almost proportionally to the noise level.

The other two tables, Tables 11 and 12, show individual and summary contributions of the first two factors to the data scatter at Gaussian 3-mixture and 5-mixture data generators, respectively.

Obviously, contribution of the first factor at either of the latter tables is always greater than that in Table 10 at the level of noise $\epsilon = 0.6$. Yet the unidimensional global least-squares methods NIPALS and IMLS-1 outperform all the other methods at the latter's data generation model (as shown in Table 3), which clearly shows that the contribution is not valid as a data complexity measure in this case.

6. Conclusion

In this work, we summarised a number of least-squares data imputation techniques that extend the singular value decomposition of complete data matrices to the case of incomplete data. It appears, there are two principal approaches to this: (1) by iteratively fitting the available data only, as in ILS, and (2) by iteratively updating a completed data matrix, as in IMLS.

Then we proposed local versions of the IMLS and ILS methods based on utilising the nearest neighbour (NN) approach. Also, a combined method INI has been developed by using the NN approach at a globally imputed matrix.

A set of eight least-squares based methods then have been tested at simulated data to compare their performances. The well-known average scoring method Mean and its NN version, N-Mean, recently described in the literature, have been used as the bottom-line.

The results show that the relative performances of the methods depend on the character of data and missings. In particular, global methods perform well in the situations in which the data are generated from a unidimensional source, even when the level of noise is significant. With a more complex Gaussian mixture data structure the local versions perform better. Even N-Means may do relatively well at a complex data structure when missings are rare. However, the only method to consistently outperform the others, especially at the data of complex structure with a proportion of missings in the range of 5–25 per cent, is the combined method INI.

With regard to the issues raised, directions for future work should include the following:

- (1) It should be a theoretical investigation carried out on the properties of convergence for both major iterative techniques, ILS and IMLS. In our computations, ILS does not always converge, even when the Gabriel–Zamir setting is applied to initialise the process.

- (2) The performances of the least-squares based techniques should be compared with those of another set of popular imputation techniques based on the maximum likelihood principle. This requires some work since the latter methods are computationally expensive and may fail to converge. In our preliminary experiments, though, the errors were similar between the global least-squares imputation techniques and standard expectation–maximization techniques implementing the maximum likelihood principle.
- (3) A measure of data complexity should be introduced to guide the user in selecting data-specific imputation methods. However, this is not an easy task: it appears, a seemingly good measure, the factor contribution to the data scatter, cannot do the job.
- (4) The simulation design should be further advanced in both the variety of data structures and missing patterns. In particular, more specific models for mechanisms of missings should be introduced and explored.

References

- [1] D. Aha, Editorial, *Artificial Intelligence Review* 11 (1997) 1–6.
- [2] S. Ahmad, V. Tresp, Some solutions to the missing feature problem in vision, in: *Advances in Neural Information Processing Systems 5*, Morgan Kaufmann, San Mateo, 1993, pp. 1712–1719.
- [3] C.G Atkeson, A.W. Moore, S. Schaal, Locally weighted learning, *Artificial Intelligence Review* 11 (1997) 11–73.
- [4] A. Christofferson, The one component model with incomplete data, PhD Thesis, Uppsala University, 1970.
- [5] P. Davies, P. Smith, *Model Quality Reports in Business Statistics*, ONS, UK, 1999.
- [6] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society* 39 (1977) 1–38.
- [7] R. Dybowski, Classification of incomplete feature vectors by radial basis function networks, *Pattern Recognition Letters* 19 (1998) 1257–1264.
- [8] EM based imputation software. Available from <<http://www.stat.psu.edu/jls/misoftwa.html>, <http://methcenter.psu.edu/EMCOV.html>>.
- [9] B.S. Everitt, D.J. Hand, *Finite Mixture Distributions*, Chapman and Hall, 1981.
- [10] K.R. Gabriel, S. Zamir, Lower rank approximation of matrices by least squares with any choices of weights, *Technometrics* 21 (1979) 298–489.
- [11] Generation of Gaussian mixture distributed data, NETLAB neural network software. Available from <<http://www.ncrg.aston.ac.uk/netlab>>.
- [12] B. Grung, R. Manne, Missing values in principal component analysis, *Chemometrics and Intelligent Laboratory System* 42 (1998) 125–139.
- [13] T. Hastie, R. Tibshirani, G. Sherlock, M. Eisen, P. Brown, D. Botstein, Imputing missing data for gene expression arrays, Technical Report, Division of Biostatistics, Stanford University, 1999.
- [14] W.J. Heiser, Convergent computation by iterative majorization: theory and applications in multidimensional analysis, in: W.J. Krzanowski (Ed.), *Recent Advances in Descriptive Multivariate Analysis*, Oxford University Press, 1995, pp. 157–189.
- [15] I.T. Jolliffe, *Principal Component Analysis*, Springer-Verlag, 1986.

- [16] L. Kamakashi, S.A. Harp, T. Samad, R.P. Goldman, Imputation of missing data using machine learning techniques, in: E. Simoudis, J. Han, U. Fayyad (Eds.), *Second International Conference on Knowledge Discovery and Data Mining*, Oregon, 1996, pp. 140–145.
- [17] N. Kenney, A. Macfarlane, Identifying problems with data collection at a local level: survey of NHS maternity units in England, *British Medical Journal* 319 (1999) 619–622.
- [18] H.A.L. Kiers, Weighted least squares fitting using ordinary least squares algorithms, *Psychometrika* 62 (1997) 251–266.
- [19] S. Laaksonen, Regression-based nearest neighbour hot decking, *Computational Statistics* 15 (2000) 65–71.
- [20] R.J.A Little, D.B Rubin, *Statistical Analysis with Missing Data*, John Wiley and Sons, 1987.
- [21] B. Mirkin, *Mathematical Classification and Clustering*, Kluwer Academic Publishers, 1996.
- [22] T.M Mitchell, *Machine Learning*, McGraw-Hill, 1997.
- [23] I. Myrtveit, E. Stensrud, U.H. Olsson, Analyzing data sets with missing data: an empirical evaluation of imputation methods and likelihood-based methods, *IEEE Transaction on Software Engineering* 27 (2001) 999–1013.
- [24] J.R Quinlan, Unknown attribute values in induction, *Sixth International Machine Learning Workshop*, New York, 1989.
- [25] S. Roweis, EM algorithms for PCA and SPCA, in: M. Jordan, M. Kearns, S. Solla (Eds.), *Advances in Neural Information Processing Systems*, 10, MIT Press, 1998, pp. 626–632.
- [26] D.B. Rubin, *Multiple Imputation for Nonresponse in Surveys*, John Wiley & Sons, 1987.
- [27] D.B. Rubin, Multiple imputation after 18+ years, *Journal of the American Statistical Association* 91 (1996) 473–489.
- [28] J.L. Schafer, *Analysis of Incomplete Multivariate Data*, Chapman and Hall, 1997.
- [29] H.Y Shum, K. Ikeuchi, R. Reddy, PCA with missing data and its application to polyhedral object modelling, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17 (1995) 854–867.
- [30] M.E. Tipping, C.M. Bishop, Probabilistic principal component analysis, *Journal of the Royal Statistical Society Series B* 61 (1999) 611–622.
- [31] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Hastie, R. Tibshirani, D. Botstein, R.B. Altman, Missing value estimation methods for DNA microarrays, *Bioinformatics* 17 (2001) 520–525.
- [32] H. Wold, Estimation of principal components and related models by iterative least square, in: P.R. Krishnaiah (Ed.), *Multivariate Analysis, Proceedings of International Symposium in Dayton*, Academic Press, 1966, pp. 391–402.