Information Sciences 259 (2014) 596–610

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

# Incomplete-case nearest neighbor imputation in software measurement data [☆]

Jason Van Hulse, Taghi M. Khoshgoftaar [*]

*Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431, USA*

A R T I C L E I N F O

*Article history:*
Available online 9 January 2011

*Keywords:*
Nearest neighbor imputation
Complete-case
Incomplete-case
Software measurement data

A B S T R A C T

*k* nearest neighbor imputation (kNNI) is one of the most popular methods in empirical software engineering for imputing missing values. kNNI typically uses only complete cases as possible donors for imputation (called complete case kNNI or CCkNNI). Though it often produces reasonable results, CCkNNI is severely limited when the amount of missing data is large (and hence the number of complete cases is small). In response, a variant of CCkNNI called incomplete case *k* nearest neighbor imputation (ICkNNI) has been proposed as an attractive alternative. This work presents a detailed simulation comparing CCkNNI and ICkNNI using two different software measurement datasets. The empirical results show that using incomplete cases often increases the effectiveness of nearest neighbor imputation (especially at higher missingness levels), regardless of the type of missingness (i.e., the distribution of missing values in the data).

## 1. Introduction

The occurrence of missing values is a pervasive problem in software measurement datasets. Unfortunately, it is often impossible to obtain the actual values, and hence the data analyst must contend with the reality that some data is unavailable. To complicate matters further, many algorithms used to analyze measurement data require complete (i.e., non-missing) data in order to execute. For example, suppose the objective is to construct a regression model using past project data which predicts the number of program faults given a set of software measurements. In general, regression techniques are unable to directly deal with missing values, and hence it is necessary to handle the missing data before the regression model can be built. One common solution used by many empirical software engineering (SE) practitioners is called listwise deletion (LD). LD eliminates instances (program modules) with missing values from the data before analysis, resulting in a smaller, complete dataset. Though it is simple to implement, LD can cause significant problems, as mentioned both by researchers in empirical SE [4,11] and in general statistical literature [1]. Further, as SE datasets can be small in size, the information loss caused by LD can be intolerable.

Numerous *imputation* procedures, which fill-in or impute the missing values with one (or more) alternative values, have been proposed in an attempt to alleviate the problems caused by missing data. Imputation techniques have the advantage that no data is discarded, and that normal complete-case methods and algorithms can be used once the imputation is complete. One of the most popular imputation methods in empirical SE is *k* nearest neighbor imputation or kNNI. kNNI is often used because it is simple to implement and often provides good imputation performance. kNNI requires a case library from

☆ This is an expanded version of the work accepted and presented at the 2007 IEEE International Conference on Information Reuse and Integration (IRI) [19].

* Corresponding author. Tel.: +1 561 297 3994; fax: +1 561 297 2800.
*E-mail addresses:* jvanhulse@gmail.com (J. Van Hulse), khoshgof@fau.edu (T.M. Khoshgoftaar).

0020-0255/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2010.12.017

which the imputations are drawn, and the case library typically consists of only complete instances (we abbreviate this version of kNNI as CCkNNI). This restriction can be a severe problem, however, especially when the amount of missing data is high and there may be few (if any) complete instances. An alternate version of CCkNNI (called incomplete case kNNI or ICkNNI [4]) relaxes the restriction that all examples in the case library be complete, allowing some incomplete cases to also act as donors. The potential drawback of ICkNNI is that it is somewhat more complex and software may not be readily available.

The contribution of this work is to provide a detailed experimental comparison of CCkNNI and ICkNNI in the context of imputing missing software measurement data under various missingness scenarios. To our knowledge, ICkNNI has been the subject of only preliminary experimentation [4]. To achieve our objectives, we implemented software tools in the SAS programming language [12] to execute both CCkNNI and ICkNNI. Detailed simulations using two real-world software measurement datasets called JM1-2445 and CCCS were performed, as described in detail in Section 4. Statistical analysis of the results is provided, and they demonstrate that in almost all cases, ICkNNI is better or significantly better than CCkNNI, especially at higher levels of missingness. We therefore conclude that ICkNNI is an attractive alternative to CCkNNI for imputing missing measurement data, especially when the level of missingness is high.

The remainder of this work is organized as follows. Section 2 provides an overview of the CCkNNI algorithm, and ICkNNI is described in Section 3. The experimental design is presented in Section 4, and related work in the domain of missing values in SE datasets is presented in Section 5. Section 6 presents the experimental results. Conclusions and directions for future work are provided in Section 7. The appendices (Appendices A and B) provide detailed information on each of the datasets used in the experiments and the process of injecting missing values in those datasets.

## 2. Complete-case nearest neighbor algorithm

The algorithm for complete-case $k$ nearest neighbor imputation (CCkNNI) is provided in Fig. 1. The case library of possible nearest neighbors for each example $\mathbf{x}_i$ to be imputed is the set of complete examples $\mathcal{C}$. The distance between instance $\mathbf{x}_i \in \mathcal{M}$ with missing values and each complete example $\mathbf{x}_j \in \mathcal{C}$ is computed, and the set of $k$ nearest neighbors $\mathcal{K}_i$ is found (line 3). In other words, $\mathcal{K}_i$ is the set of $k$ complete examples that are closest, as measured by some distance measure, to the example $\mathbf{x}_i$. Each missing attribute value in $\mathbf{x}_i$ is imputed with the average value of the $k$ nearest neighbors (line 4); the imputed attribute values for instance $\mathbf{x}_i$ are denoted $\hat{x}_{im_l}$, $l = 1, \ldots, \alpha(i)$. The finally imputed example is denoted $\hat{\mathbf{x}}_i$ on line 5, where each missing attribute value is replaced by the imputed values. The procedure returns an imputed dataset. Since our dataset only contained numeric attributes, and for simplicity, we assume numeric attributes for imputation, though only simple modifications are needed to handle other data types. There are also a number of different versions of kNN imputation, which employ various attribute standardization algorithms. Simple modifications can be made to Fig. 1 to accommodate these scenarios, and the intention of this study is not to analyze all of the different versions of complete-case kNNI.

## 3. Incomplete-case nearest neighbor algorithm

The algorithm for incomplete-case $k$ nearest neighbor imputation (ICkNNI) is provided in Fig. 2. The main difference between ICkNNI and CCkNNI is that the set of possible nearest neighbors for an example $\mathbf{x}_i$ being imputed *changes depending on the attribute value being imputed*. In particular, the requirement that all examples in the case-library be complete is relaxed. Instead, if attribute $x_j$ of instance $\mathbf{x}_i$ is being imputed, then eligible nearest neighbors are those examples $\mathbf{x}_l$ which have the same subset of observed attributes as $\mathbf{x}_i$, and $x_{lj}$ is also observed (line 3). The set of *potential* nearest neighbors when imputing

**Procedure: Complete-Case kNN imputation**
**input**: Dataset $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{n}$, where $\mathbf{x}_i = (x_{im_1}, \ldots, x_{im_{\alpha(i)}}, x_{io_1}, \ldots, x_{io_{\beta(i)}})$ has $\alpha(i)$ missing attribute values $x_{im_1}, \ldots, x_{im_{\alpha(i)}}$ and $\beta(i)$ observed values $x_{io_1}, \ldots, x_{io_{\beta(i)}}$. Further, $\mathcal{D} = \mathcal{C} \cup \mathcal{M}$, where $\mathcal{C}$ is the set of complete examples ($\mathcal{C} = \{\mathbf{x}_j \in \mathcal{D} \mid \alpha(j) = 0\}$) and $\mathcal{M} = \mathcal{D} \setminus \mathcal{C}$; number of nearest neighbors $k$. The $w$ attributes are denoted $x_1, \ldots, x_w$, and we assume $w = \alpha(i) + \beta(i)\ \forall\ \mathbf{x}_i \in \mathcal{D}$.
**output**: Dataset $\widehat{\mathcal{D}} = \mathcal{C} \cup \widehat{\mathcal{M}}$ where $\widehat{\mathcal{M}}$ is the set of imputed examples.

```
1. do $\forall\ \mathbf{x}_i \in \mathcal{M}$:
2.     $\forall\ \mathbf{x}_j \in \mathcal{C}$, find $d_j = d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{l=1}^{\beta(i)} (x_{io_l} - x_{jo_t})^2}$ where $o_t = o_l$.
3.     Let $\mathcal{K}_i = \{\mathbf{x}_j \in \mathcal{C} \mid d_j \leq d_q\ \forall\ \mathbf{x}_q \notin \mathcal{K}_i\}$; $k = \mid \mathcal{K}_i \mid$.
4.     For $l = 1, \ldots, \alpha(i)$, $\hat{x}_{im_l} = k^{-1} \sum_{\mathbf{x}_p \in \mathcal{K}_i} x_{po_t}$; $o_t = m_l$.
5.     $\hat{\mathbf{x}}_i = (\hat{x}_{im_1}, \ldots, \hat{x}_{im_{\alpha(i)}}, x_{io_1}, \ldots, x_{io_{\beta(i)}})$.
6. end do
7. $\widehat{\mathcal{M}} = \{\hat{\mathbf{x}}_i \mid \mathbf{x}_i \in \mathcal{M}\}$, $\widehat{\mathcal{D}} = \mathcal{C} \cup \widehat{\mathcal{M}}$.
8. Return $\widehat{\mathcal{D}}$.
```

**Fig. 1.** Complete-case kNN imputation.