Handling Missing Values via Decomposition of the Conditioned Set

Mei-Ling Shyu, Indika Priyantha Kuruppu-Appuhamilage
Department of Electrical and Computer Engineering, University of Miami
Coral Gables, FL 33124, USA
shyu@miami.edu, ikuruppu@umsis.miami.edu

Shu-Ching Chen
Distributed Multimedia Information System Laboratory, School of Computer Science
Florida International University, Miami, FL 33199, USA
chens@cs.fiu.edu

LiWu Chang
Center for High Assurance Computer Systems
Naval Research Laboratory, Washington, DC 20375, USA
lchang@itd.nrl.navy.mil

Abstract

In this paper, a framework for replacing missing values in a database is proposed since a real-world database is seldom complete. Good data quality in a database can directly improve the performance of any data mining algorithm in various applications. Our proposed framework adopts the basic concepts from conditional probability theories and further develops an algorithm to facilitate the capability of handling both nominal and numerical values, which addresses the problem of the inability of handling both nominal and numerical values with a high degree of accuracy in the existing algorithms. Several experiments are conducted and the experimental results demonstrate that our framework provides a high accuracy when compared with most of the commonly used algorithms such as using the average value, using the maximum value, and using the minimum value to replace missing values.

1 Introduction

The performance of any data mining application heavily depends on the quality of the data in the database, where data quality refers to the accuracy and completeness of the data. Especially, the databases in practical applications are usual large, where the problems of inaccurate and/or inconsistent data are inevitable [1]. In a real-world environment, there are many possible reasons why the inaccurate or inconsistent data occur in a database, e.g., equipment malfunctioning, the deletion of data instances (or records) due to the inconsistency with other recorded data, not entering data due to misunderstanding, considering the data as unimportant at the time of entry, etc. This may cause numerous missing values in the database, which can negatively impact the discovered patterns/knowledge from the data mining algorithm. Moreover, the errors or data skews can even proliferate across subsequent runs, which causes a larger and cumulative error effect. Hence, data preprocessing (particularly missing value handling) that describes any types of processes performed on the raw data to prepare them for other processing procedures is needed.

Data preprocessing usually involves cleaning the data before they are used in data mining tools [9][11] or any applications. The need for data cleaning increases significantly especially when there are multiple data sources that need to be integrated [2][4]. Thus, data cleaning has become a vital and challenging aspect in preprocessing [5][7][8]. For example, some issues and current approaches in data cleaning in a single source and multiple sources were discussed in [9]. Missing value handling is one of the major tasks in data cleaning.

Numerous techniques have been developed to handle missing values in the literature. For example, a missing value can be ignored, which is the easiest way to handle missing values, but it does not contribute to enhance the quality of the database that has the missing values [10][12]. Another typical way to handle missing values is to replace them by using the average value, substituting with the maximum value, or using the minimum value. However, these

techniques can only be applied when the attribute consists of numeric values because the statistical measures used are defined only on numerical values. Inferring the most probable value to fill in the missing value (a crude version of the density-based model) is also used [6]. However, this approach is useful only when the attribute values are nominal since it is difficult to define the most probable value for a set of continuous values. On the other hand, the method of considering the missing value as another distinct value works on both numerical and nominal values.

To address this issue, a novel framework called F-DCS for replacing missing values is proposed in this paper. F-DCS stands for the Framework based on the Decomposition of the Conditioned Set, which adopts the concepts on conditional probability and is based on the decomposition of the conditioned set. It has several advantages. First, it has the capability of handling both numerical and nominal missing attribute values with a comparatively higher percentage of accuracy when compared with the other existing techniques. Second, when considering the method of using the most frequent value, the replacing value is taken as the mode of all the values in that particular attribute. However, our framework refers only to the most relevant set of values to estimate the missing value. Thus, fewer ambiguities are introduced in selecting the value for replacing the missing values. Third, unlike the other methods such as using the average value or using the mean value that perform better only when the data fits into a normal distribution, our proposed framework is not restricted to any distributional assumptions. Finally, our framework does not need to be trained, and is built/developed using the same input data set. Therefore, the necessity of a training data set can be avoided, which makes our framework best fits into real-world applications.

We have conducted a set of experiments on three databases at the UCI Machine Learning Repository [13] to evaluate our proposed F-DCS framework. The experimental results demonstrate that the proposed framework can handle both nominal and numerical values with a high degree of accuracy when compared with any other techniques. The results also show our superiority with a high degree of certainty.

This paper is organized as follows. Section 2 describes our proposed F-DCS framework in details. The experimental procedure together with the experimental results are presented in Section 3. We conclude our study in Section 4.

2 Framework for Handling Missing Values

In this section, our proposed framework for handling missing values based on the concepts of conditional probability theories is presented. The F-DCS framework uses the decomposition of the conditioned set starting from the maximum number of elements to accommodate for multiple missing values, and has the capability to handle both numerical and nominal missing data values.

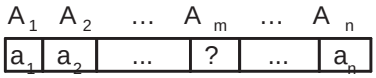

Figure 1. Data instance having a single missing value (Attribute A_m)

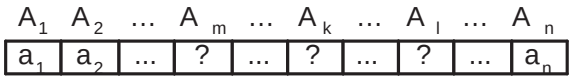

Figure 2. Data instance having multiple missing values (Attributes A_m, A_k, and A_l)

Practically, a single data instance may have one or more missing values as shown in Figure 1 and Figure 2, respectively. As can be seen from these two figures, assume the database has n number of attributes, denoted by A_i, where $1 \leq i \leq n$. Consider one data instance from the database. The value that an attribute can take is represented by a_i, where $1 \leq i \leq n$. A "?" in the corresponding entry indicates the value for that attribute is missing. For example, Figure 2 has three missing values for attributes A_m, A_k, and A_l. In replacing missing values, each missing value is considered one at a time with the following procedure.

Considering a missing value in attribute A_m, the set Q^{n-1} is defined as shown in Equation 1, where $\{Q_1 : A_1 = a_1\}$ denotes the set of data instances having a value a_1 for the attribute A_1 by considering all the data instances of the database. The condition $(A_m = a_m)$ is excluded from the set Q^{n-1}.

$$\begin{aligned} Q^{n-1} &= \{Q_1 : A_1 = a_1\} \cap \{Q_2 : A_2 = a_2\} \cap \ldots \\ &\quad \cap \{Q_{m-1} : A_{m-1} \\ &= a_{m-1}\} \cap \{Q_{m+1} : A_{m+1} = a_{m+1}\} \cap \ldots \\ &\quad \cap \{Q_n : A_n = a_n\} \end{aligned} \quad (1)$$

In our proposed framework, the idea is to match the values of the corresponding attributes between the data instance with missing values and the rest of the data instances in the data file. For nominal attribute values, the probable set of the values that attribute A_m can have is denoted by the set $\{m_1, m_2, m_3, \ldots, m_q\}$ assuming there are q possible values for A_m. The two values of a nominal attribute are said to be matched if their values are equal. For each attribute with missing values (say the m^{th} attribute), we keep a count for each probable value from all the data instances

for that attribute, and the value with the maximum count is used to replace the missing value. Section 2.1 discusses how the proposed framework handles missing values that are nominal. On the other hand, such a set of probable values cannot be defined for numerical attribute values, and it is difficult to have an exact match. Therefore, a different approach that introduces a threshold value to take care of this problem is proposed, and will be discussed in Section 2.2.

2.1 Nominal Attributes

Assume that the set M contains q subsets $(M_1, M_2, \ldots, M_q)$ as given in Equation 2. Let Θ be the set of all possible mutually exclusive outcomes of an experiment, Σ be an algebra defined on Θ, and $P(\Theta, \Sigma)$ be the set of all probability measures on Σ. According to the conditional probability theory [3], when each subset in $(M_1, M_2, \ldots, M_q)$ is considered at a time as the set B, $P(.) \in P(\Theta, \Sigma)$ and $P(Q^{n-1}) > 0$ for some $Q^{n-1} \subset \Sigma$, the conditional probability of B given Q (i.e., $P(B \mid Q^{n-1})$) can be defined in Equation 3. This will create a set W consisting of q probabilities by calculating the frequency count. Let $1 \leq j \leq q$ and $w_j = P(M_j \mid Q^{n-1})$, the set W is given in Equation 4. From the set W, the maximum probability can be found, say w_j. Therefore, the missing value for attribute A_m is replaced by m_j, where m_j corresponds to the set B that satisfies the condition $(A_m = m_j)$.

$$\begin{aligned} M &= \{M_1 : A_m = m_1\} \cup \{M_2 : A_m = m_2\} \cup \ldots \\ &\cup \{M_j : A_m = m_j\} \cup \ldots \{M_q : A_m = m_q\}. \quad (2) \end{aligned}$$

$$P(B \mid Q^{n-1}) = \frac{P(Q^{n-1} \cap B)}{P(Q^{n-1})}. \quad (3)$$

$$W = \{w_1, w_2, \ldots, w_j, \ldots, w_q\}. \quad (4)$$

However, it is possible that a non-zero probability in any element of the set W cannot be found. In such a scenario, the concept of decomposition of the conditioned set is utilized. Define the set Q^{n-2} by removing one subset at a time from the set Q^{n-1}. The set Q^{n-2} becomes a superset of Q^{n-1} assuming that the first subset is removed from the set Q^{n-1}.

$$\begin{aligned} Q^{n-2} &= \{Q_2 : A_2 = a_2\} \cap \ldots \{Q_{m-1} : A_{m-1} = a_{m-1}\} \\ &\cap \{Q_{m+1} : A_{m+1} = a_{m+1}\} \cap \ldots \{Q_n : A_n = a_n\} \end{aligned}$$

Using the set Q^{n-2} as the conditioned set, the new set of probabilities for W can be calculated. Since there are (n-1) ways to remove one subset from Q^{n-1} to generate the superset Q^{n-2}. Therefore, the conditional probabilities should be calculated by considering each superset as the conditioned set, which results in $(n-1) \times q$ probability values. If a non-zero probability in any of the element in W still cannot be found, two subsets from the set Q^{n-1} are removed to generate the superset Q^{n-3}. Continue such a procedure until at least one non-zero probability value in W is found. However, the number of subsets that can be removed from Q^{n-1} becomes larger and larger, the total number of supersets that can be generated becomes large as follows.

$$(n-1) + (n-1)(n-2) + (n-1)(n-2)(n-3) + \ldots + \ldots + (n-1)(n-2)\ldots 2$$

Though this number seems to be large, however, time complexity is not really a major concern in this study. The reasons are that (i) all the probabilities can be calculated in one database scan for a particular missing value, and (ii) the data cleaning process is usually done off-line.

2.2 Numerical Attributes

In our proposed framework, a technique for using a threshold value to deal with the numerical values has been introduced instead of discretizing the values. The advantage of using a threshold value over using discretization is that the threshold value indicates some distance measure between two values; while discretization converts each value to a pre-set partition, which does not reflect the actual distance between two values. As an example, consider a set of integers from 1 to 10, and assume that the number of partitions in discretization is 5. Then the numbers 1 and 2 will be assigned to the first partition, the numbers 3 and 4 go to the second partition, and so on. When the numbers 1 and 2 are considered, they are similar because they are in the same partition. On the other hand, in calculating the threshold value, a threshold value can be set in the way that these two numbers can be considered as similar. When the numbers 2 and 3 are considered, using the same threshold value, they can be considered as similar, too. This process is more reasonable because the difference between these two numbers is the same as that in the first case. However, in generic discretization, these two numbers will go to different partitions and cannot be considered as similar. Therefore, using the threshold value approach proposed in our framework gives a better indication of the distance between two numbers.

Let any two numerical values, n_1 and n_2, belong to the same attribute, and let "range" denote the difference between the maximum and minimum values for that particular attribute, n_1 and n_2 are considered to be equal when their *Thresholdvalue* (calculated from Equation 5) exceeds a pre-set threshold value. From empirical studies, the value *20* is considered as the pre-set threshold value in the conducted experiments.

$$Thresholdvalue = | \frac{(n_1 - n_2)}{range} | \times 100. \quad (5)$$

Unlike in the nominal case, the maximum probability for each candidate value for a missing value cannot be calculated since the attribute value is continuous. Hence, the average value of the selected set of candidates for the missing value is used. When the equal probable values come across for a particular missing value, the ties are broken by considering the most frequent value for that particular attribute.

3 Experimental Result Analysis

In order to evaluate the performance and accuracy of our proposed F-DCS framework in replacing missing values for the attribute(s), several experiments using three databases at the UCI Machine Learning Repository [13] are conducted. These three databases were selected in the manner that one has only numeric attributes, one has only nominal attributes, and one has both numerical and nominal attributes to demonstrate our proposed framework can handle both the nominal and numerical attributes.

Table 1 shows the databases that were used in the experiments. As can be seen from this table, the *Ionosphere* database consists of 34 attributes that are all numerical and have both positive and negative values. The *Cylinder Bands* database has the mix of nominal (19) and numerical (20) attributes. The *Mushrooms* database, on the other hand, has only 22 nominal attributes.

Table 1. UCI databases used in experiments

Database Name	Nominal Attributes	Numerical Attributes
Ionosphere Database	0	34
Cylinder Bands Database	19	20
Mushrooms Database	22	0

For comparison purposes, for each database, ten sample files are first generated and then used as the reference data files in calculating the accuracy of replacing the missing values. For each sample file, another set of files having the missing value percentages of 2%, 5%, 10%, 20%, 25%, 50%, and 70% are generated by artificially introducing missing values into the database by distributing them randomly and uniformly into the database, since the databases originally have no missing. However, a database may contain a certain low percentage of missing values if it is to be used for some applications. Therefore, in selecting the missing value percentages, more concern is placed on the lower percentage of missing values (for example, 2%, 5%, 10%, 15%, 20%, and 25%). The percentages of 50% and 70% are used for the completeness of the performance evaluation purpose, although it is almost worthless to use any database consisting of more than 50% missing values in practical applications.

Assume each file is represented as an $m \times n$ matrix (i.e., m rows and n columns), and let r_1 and r_2 be two random numbers such that $1 \leq r_1 \leq m$ and $1 \leq r_2 \leq n$. Then, a missing value is inserted to (r_1, r_2) entry in the file. Continuing the same procedure, the missing values are introduced until the required missing value percentage of the file is reached. Let $N_{missing}$ be the number of missing values in the file, the missing value percentage is calculated as follows.

$$\text{Missing-value-percentage} = \frac{N_{missing}}{m \times n} \times 100.$$

After generating all the files with the missing values, the F-DCS framework is compared with the commonly used techniques such as using the average value, using the maximum value, and using the minimum value to replace missing values. They are denoted by "F-DCS", "Avg", "Max", and "Min", respectively in the experimental results.

An important issue to be addressed is that all these techniques are only defined for numerical values. Therefore, a different consideration needs to be used when dealing with nominal attributes. That is, the most frequent value for a nominal attribute is chosen to replace the missing value. Thus, for the *Ionosphere* and the *Cylinder Bands* databases, all the three techniques are used to compare with our F-DCS framework. On the other hand, for the *Mushroom* database, only the average technique that uses the most frequent values is used in the comparison. The performance evaluation is to compare each resulting file with the corresponding reference file and to calculate the percentage of the accurately filled values for those missing values. Each combination of a database and a missing value percentage, the percentages of the accuracy for the ten sample files are calculated and their average accuracy is reported in the performance comparison.

3.1 Performance Comparison for the *Ionosphere* Database

The first comparison is shown in Table 2 for the *Ionosphere* database (consisting of both negative and positive numerical values). From this table, it can be easily seen that our framework achieves above 80% accuracy for the missing value percentages less than 50%, and outperforms any other techniques under the various missing value percentages. On the other hand, the average value method achieves around 60% accuracy, and the maximum value method and the minimum value method both have very low accuracy performance.

The first column of Table 2 describes the missing value percentages of the input files. As an example, when the

average accuracy of our F-DCS framework is 86.72% when the input files have 2% missing values; while the average accuracies are 58.86%, 39.36%, and 12.35% respectively for the methods of using the average value, using the maximum value, and using the minimum value under the same missing value percentage. This demonstrates that our framework constantly replaces the missing values with a high degree of accuracy; while the methods of using the minimum value and using the maximum value achieve very low accuracy values.

Table 2. Accuracy of filling missing values for the *Ionosphere* Database

Percent	F-DCS	Average	Maximum	Minimum
2%	86.72%	58.86%	39.36%	12.35%
5%	85.01%	56.91%	37.02%	12.95%
10%	85.35%	59.07%	36.83%	11.96%
15%	84.79%	59.51%	36.65%	12.57%
20%	83.81%	57.95%	36.37%	12.89%
25%	83.38%	57.69%	36.86%	12.78%
50%	78.36%	57.98%	36.52%	12.82%
70%	70.43%	57.61%	36.54%	12.89%

3.2 Performance Comparison for the *Cylinder Bands* Database

The second experiment is to compare the performance when a database has both numerical attributes and nominal attributes. For this purpose, the *Cylinder Bands* database that has almost the same number of numerical and nominal attributes is considered. The original database consists of 2% missing values. Therefore, there is no row for the 2% case in Table 3. Also, since this database has both types of attributes, we have to modify all the techniques to facilitate their capabilities in handling such a scenario by selecting the most frequent value to deal with nominal values as described earlier in this section.

As can be seen from Table 3, the F-DCS framework outperforms the other techniques for most of the missing value percentages (5%, 10%, 20%, 25%, and 50%) with an exception of the 70% of missing values, where the average technique performs better than the F-DCS framework. When the missing value percentages are below 70%, the F-DCS framework outperforms the other methods by achieving above 80% of accuracy. Please note that a database having more than 50% of missing values is practically considered as obsolete.

Table 3. Accuracy of filling missing values for the *Cylinder Bands* Database

Percent	F-DCS	Average	Maximum	Minimum
5%	83.37%	71.81%	41.37%	41.34%
10%	82.88%	72.04%	37.75%	40.04%
15%	81.33%	73.79%	37.93%	41.54%
20%	79.72%	73.50%	38.74%	42.63%
25%	78.20%	73.15%	37.96%	41.58%
50%	71.97%	71.13%	38.81%	40.81%
70%	67.69%	69.35%	39.76%	41.27%

3.3 Performance Comparison for the *Mushrooms* Database

The third experiment is conducted on the *Mushrooms* database that has only the nominal attributes, unlike the other databases that consist of numerical attributes. Similarly, it is selected to demonstrate the F-DCS framework has the capability to handle both numerical and nominal attribute values. Since this database has only nominal attributes, it does not seem to be meaningful to compare the F-DCS framework with the techniques of using the maximum value and using the minimum value. Hence, only the modified version of the average technique is used to compare its results with the F-DCS framework.

Table 4 clearly shows that the F-DCS framework handles the nominal attribute values much better than the average technique. For all the missing value percentages, the F-DCS framework always provides more than 70% accuracy but the average technique can only reach less than 60% accuracy.

Table 4. Accuracy of filling missing values for the *Mushrooms* Database

Percent	F-DCS	Average
2%	70.05%	58.80%
5%	73.29%	58.82%
10%	76.55%	58.98%
15%	78.00%	59.08%
20%	78.56%	59.06%
25%	78.82%	58.96%
50%	76.82%	58.99%
70%	70.53%	58.92%

From the results of these three experiments, it can be shown that the F-DCS framework can stably and consistently achieve relatively high accuracy values for various percentages of missing values and for databases with only

numerical attributes, only nominal attributes, and both numerical and nominal attributes.

4 Conclusion

In this paper, a novel framework, called F-DCS, based on the decomposition of the conditioned set from the conditional probability for replacing missing values as a data cleaning tool is proposed. A good data cleaning tool can improve the performance of the data mining algorithms for various applications. A set of experiments on three databases in the UCI repository are conducted to evaluate the performance of the F-DCS framework. The selected databases include one with purely nominal attributes, one with purely numerical, and one with mixed numerical and nominal attributes. From the experimental results, it can be concluded that the F-DCS framework has the capability to handle both nominal and numerical values with a high degree of accuracy when compared with the other commonly used techniques such as using the average value, using the maximum value, and using the minimum value. The proposed F-DCS framework achieves almost above 80% accuracy for lower percentages of missing values, which are the most common scenarios for any real-world application.

5. Acknowledgment

For Mei-Ling Shyu, this research was supported in part by NSF ITR (Medium) IIS-0325260. For Shu-Ching Chen and Mei-Ling Shyu, this research was supported in part by Naval Research Laboratory (NRL)/ITT: 176815J.

References

[1] D. P. Ballou and G. K. Tayi, "Enhancing Data Quality in Data Warehouse Environments," *Communications of ACM*, 42, 1999, pp. 73–78.

[2] M. G. Ceruti and M. N. Kamel, "Preprocessing and Integration of Data from Multiple Sources for Knowledge Discovery," *International Journal on Artificial Intelligence Tools*, 1999, pp. 157–177.

[3] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*, Second Edition, Wiley-Interscience, New York, NY, 2001.

[4] H. Galhardas, D. Florescu, D. Shasha, and E. Simon, "An Extensible Framework for Data Cleaning," In *Proceedings of the International Conference on Data Engineering (ICDE)*, San Diego, CA, 2000, pp. 312

[5] H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C. A. Saita, "Declarative Data Cleaning: Language, Model, and Algorithms," In *Proceedings of Int. Conf. on Very Large Data Bases (VLDB'01)*, Rome, Italy, 2001, pp. 371–380.

[6] R. J. Little and D. B. Rubin. *Statistical Analysis with Missing Data*, John Wiley and Sons, New York, 1987.

[7] J. I. Maletic and A. Marcus, "Data Cleansing: Beyond Integrity Analysis," In *Proceedings of the International Conference on Information Quality*, 2000, pp. 200–209.

[8] A. Maydanchik, "Challenges of Efficient Data Cleansing," *DM Review*, September Issue, 1999.

[9] E. Rahm and H. H. Do, "Data Cleaning: Problems and Current Approaches," *Bulletin of the IEEE Technical Committee on Data Engineering*, 23, 4, 2000, pp. 3–13.

[10] T. Redman. *Data Quality: Management and Technology*, Bantam Books, New York, 1992.

[11] P. Vassiliadis, Z. Vagena, S. Skiadopoulos, N. Karayannidis, and T. Sellis, "Arktos: A Tool for Data Cleaning and Transformation in Data Warehouse Environments," *Bulletin of the IEEE Technical Committee on Data Engineering*, 23, 4, 2000, pp. 42–47.

[12] Y. Wand and R. Wang, "Anchoring Data Quality Dimensions Ontological Foundations," *Communications of ACM*, 39, 1996, pp. 86–95.

[13] http://www.ics.uci.edu/ mlearn/MLSummary.html