Imputation of Missing Values for Semi-supervised Data Using the Proximity in Random Forests

Tsunenori Ishioka
The National Center for University Entrance Examinations
2-19-23 Komaba, Meguro-ku
Tokyo, Japan
tunenori@rd.dnc.ac.jp

ABSTRACT

This paper presents a procedure that imputes missing values by using random forests on semi-supervised data. We found that the rate of correct classification of our method is higher than that of other methods: a simple expansion of Liaw's "rfImpute" for (un)supervised data and the k-nearest neighbor method (kNN). Our method can handle missing predictor variables as well as missing response variable. An imputation that uses random forests for semi-supervised cases in the training data set has never been implemented until now.

Categories and Subject Descriptors

G.4 [**Mathematical Software**]: Algorithm design and analysis; I.2.6 [**Artificial Intelligence**]: Learning—*analogies*

General Terms

Algorithms

Keywords

Ensemble learning, data imputation, missing data, k-nearest neighbor, R, rfImpute, impute.knn

1. INTRODUCTION

A context is a word that has various meanings depending on the situation and/or topic. In information engineering, it indicates a situation in which a device is used. Time and place are typical examples. A user's profile and purchase history are also examples. If we can utilize the context, appropriate services can be offered according to the situation; for example, a mobile phone could recommend nearby shops to pedestrians. Various sensor information can now be easily obtained, thanks to innovation. The context information, however, is often incomplete. In many cases, the versions of sensors are diverse. Some of the information tends to get lost in a wireless environment. In fact, complete information without missing values is quite rare in the real world.

iiWAS 2012, 3-5 December, 2012, Bali, Indonesia.

The easiest to treat missing values is to "remove" them. Another way is to make a reasonable inference from the observations to account for the missing data. Here, we shall impute missing values by using random forests.

Random forests is a well-known as a substantial modification of bagging techniques. It builds a large collection of de-correlated trees and then averages them. It is mainly used as an accurate classifier or regression tree. The latest Fortran77 code by Breiman [4] is Version 5.1, dated 2004. Version 4 contains modifications and major additions to Version 3.3, and it allows one to replace missing predictor values through two options [3]. One is "missquick" (Ver. 4), which replaces all missing values by the median of the non-missing values in their column, if they are real, or by the most numerous value in their column if they are categorical. Another is "missright" (Ver. 5). This option starts with "missquick" but then iterates by using proximities and does an effective replacement even with a large amount of missing data. Missing values are represented by a proximity weighted sum over the non-missing values.

Andy Liaw implemented these ideas in statistical environment R [10], calling them "na.roughfix" and "rfImpute" [8]. These R functions unfortunately cannot be used in semi-supervised cases [5].

However, Breiman's ideas can be extended to semi-supervised data if we can obtain the proximity of the semi-supervised data, by starting with the rough imputation of the missing data ("na.roughfix") and repeatedly getting new proximities by running random forests. Another imputation method called "yaImpute" [9], which is an extended random forests, is also not allowed for semi-supervised data, because it devised an experimental way to handle multi-response variables. In the case of supervised data, Breiman [3] found that the estimation error of the bootstrap training sample (called "out-of-bag", or oob) tends to be optimistic when random forests is run on a data matrix with imputed values.

In this paper, we present a procedure for properly imputing missing values, that can avoid overfitting of the estimated model for semi-supervised data. In Section 2, we summarize the elements of a technique that imputes the missing values for semi-supervised data. In Section 3, we show the new procedure for imputing them. Section 4 illustrates two examples, iris and spam data sets. We assume these data to be semi-supervised by ignoring part of the response variables; nevertheless, both are supervised. Section 5 concludes the paper.

2. RFIMPUTE

2.1 Proximity measure

The (i, j) element of the proximity matrix produced by a random forest is the fraction of trees in which elements i and j fall on the same terminal node. The intuition is that "similar" observations should be on the same terminal nodes more often than dissimilar ones. The proximity matrix can be used to identify structures in the data and for semi-supervised learning with random forests.

2.2 R procedure

Missing values are indicated by NAs in R [10]. A function returning a result of random forests is "randomForest," developed by Liaw [8]. The algorithm starts by imputing NAs by using "na.roughfix." Then, "randomForest" is called with the completed data. The proximity matrix output from the "randomForest" is used to update the imputation of the NAs. For continuous predictors, the imputed value is the weighted average of the non-missing observations, where the weights are the proximities. For categorical predictors, the imputed value is the category with the largest average proximity. This process is iterated a few times.

3. NEW PROCEDURE

3.1 Missing value replacement for the training set

Our procedure, as well as Liaw's "rfImpute," has two ways of replacing missing values. The first way is fast. If the mth variable is not categorical, the method computes the median of all values of this variable in class j; then it uses this value to replace all missing values of the mth variable in class j. If the mth variable is categorical, the replacement is the most frequent non-missing value in class j. These missing values are replaced or filled by "na.roughfix."

The second way of replacing missing values is computationally more expensive but performs better than the first, even with large amounts of missing data. It begins by doing a rough and inaccurate filling in of the missing values. The key technique is to estimate the missing values on the basis of not all non-missing proximities but the k-nearest proximities, which include missing data. Then, it runs a forest procedure and computes the proximities.

If $x(n, m)$ is a missing continuous value, we estimate its fill as an average over the k-nearest neighbor values of the mth variables weighted by the proximities between the nth case and the other case. If it is a missing categorical variable, we replace it by the most frequent non-missing value, where the frequency is weighted by proximity.

In summary, in the case of a missing continuous value, we use

$$\widehat{x}(n,m) = \frac{\sum_{i \neq n; i \in \text{neighbor}} \text{prox}(i,n) x(i,m)}{\sum_{i \neq n; i \in \text{neighbor}} \text{prox}(i,n)} \tag{1}$$

instead of rfImpute's

$$\widehat{x}(n,m) = \frac{\sum_{i \neq n; i \in \text{non-missing}} \text{prox}(i,n) x(i,m)}{\sum_{i \neq n; i \in \text{non-missing}} \text{prox}(i,n)}$$

where $\text{prox}(\cdot, \cdot)$ is the proximity.

In the case of a missing categorical variable, we use

$$\widehat{x}(n,m) = \underset{C_m}{\text{argmax}} \sum_{i \neq n} \text{prox}(i,n), \tag{2}$$

instead of

$$\widehat{x}(n,m) = \underset{C_m}{\text{argmax}} \sum_{\substack{i \neq n \\ i \in \text{non-missing}}} \text{prox}(i,n),$$

where C_m means the mth categorical variables.

Now, we iteratively construct a forest again by using these newly filled-in values, find new fills, and iterate again. Our experience is that 4–6 iterations are enough.

The reason we use only k-nearest neighbor data in (1) is that it makes the missing data imputation statistically robust. Even if the proximities to the target are rather short, other continuous values might be outlying. In this case, some outliers will push the estimate of the target in the wrong direction. Our numerical investigation shows that our procedure, a mixture of kNN and random forests, is better than using only random forests.

In (2), however, all data besides k-nearest neighbor data are treated. Because majority votes are used in (2), any outlying values of x would be ignored. While, we should not ignore the proximity associated with missing data, especially when the missing rate is high.

3.2 Algorithm

Semi-supervised learning ought to treat missing response variable (y) as a training data set. Since our method is for the purpose of immediate imputation of missing data, both predictor variables (x) and response variable (y) can include missing values. Semi-supervised learning, which includes large amounts of missing predictor variables (x) and missing response variables (y), has a potential to cover the real world data considerably. The good semi-supervised learning method gives us many benefits. Our procedure is as follows.

1. By starting with a rough imputation of missing predictor variables (x), we estimate the missing response variable $(\hat{y})$ by running the random forests algorithm.
2. We replace the missing predictor variables $(\hat{x})$ by using the proximities between cases and estimate the response variable $(\hat{y})$.
3. If the imputed values $(\hat{x})$ converge, we output them $(\hat{x}, \hat{y})$.

We call this procedure "rfImput.smspvsd," which means "an imputation method using random forests for semi-supervised learning." We found that iterating 2 does not produce a significant improvement.

Our R program can be viewed at `http://www.rd.dnc.ac.jp/~tunenori/rfImpute.html` .

4. NUMERICAL EXAMPLES

4.1 E-mail database indicating spam or non-spam

We used a spam data set [11] collected at Hewlett-Packard Labs, which classifies 4601 e-mails as spam or non-spam. In addition to this class label, there are 57 variables indicating

the frequency of certain words and characters in the e-mail. That is, a data frame had 4601 observations and 58 variables. The first 48 variables contain the frequency of the variable name (e.g., business) in the e-mail. If the variable name starts with num (e.g., num650), it indicates the frequency of the corresponding number (e.g., 650). Variables 49–54 indicate the frequency of the characters ";", "(", "[", "!","$", and "#". Variables 55–57 contain the average, longest, and total run-length of capital letters. Variable 58 indicates the type of mail and is either "nonspam" or "spam," i.e., unsolicited commercial e-mail.

The data set contains 2788 e-mails classified as "nonspam" and 1813 classified as "spam." The "spam" concept is diverse: advertisements for products/web sites, make money fast schemes, chain letters, pornography, and so on. This collection of spam e-mails came from the collectors' postmaster and individuals who had filed spam. The collection of non-spam e-mails came from work and personal e-mails, and hence, a personal name like "george" or an area code like "650" is an indicator of non-spam. We had to partially blind the spam/non-spam indicator, because we want to focus on semi-supervised data in this numerical experiment.

To evaluate the performance of "rfImput.smspvsd," we compared it with the following two methods.

1. Liaw's "rfImpute" [5]: Since "randomForest" does not work for any y that includes missing responses, "rfImpute" functions as well. Therefore, we configured the forest model for non-missing response cases (y) by using "rfImpute" to obtain imputed predictor variables ($\hat{x}$). Using this model, we estimated the response values ($\hat{y}$) for their missing y.

2. kNN [7]: We started the rough imputation of $\hat{x}$ for y that were not missing, and configured a training kNN model with them. Using this model, we estimated the response values ($\hat{y}$) for their missing y.

Semi-supervised as well as supervised learning predict or estimate y from x. Therefore, as a criterion for evaluating the performance of learners, we used precision, that is, the rate of the correct classifications.

The values of 57 variables and response variable were randomly dropped. The missing values rates were 5%, 10%, 20%, 30%, 40%, 50%, and 60% both. Even at a low missing data rate, e.g., 5%, a complete case, which does not include missing data, is rare; the occurrence probability is only $(0.95)^{57} \approx 0.0537$. The missing data rate of 10% in turn, yields an occurrence of 0.00246. If we use only complete cases by removing missing data, almost all cases should be avoided.

Figure 1 shows the average of the results of three methods run three times each. A larger value on the vertical axis indicates better performance. A value of 1 means that all missing y are correctly predicted.

In general, the higher the missing data rate on the horizontal axis is, the smaller the value on the vertical axis becomes. Because of the randomization of the missing data, the lines on the graph do not always decrease monotonously. Nevertheless, our method ("rfImput.smspvsd") is the best of the three, irrespective of the missing data rate. Moreover, kNN is the worst because observations in high-dimensional spaces tend to be dissimilar to each other as a result of the "curse of dimensionality." In particular, the advantage of our

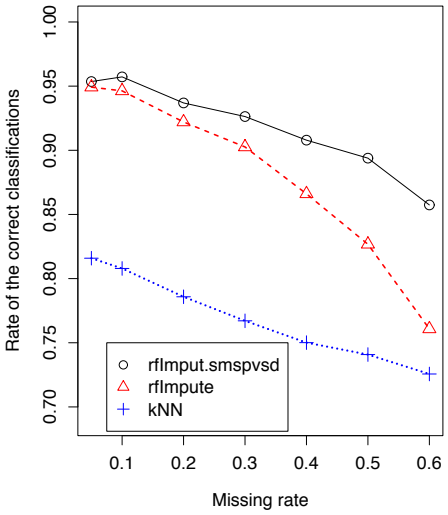

Figure 1: Correct classification for semi-supervised spam/non-spam data

method over the others is remarkable for the high missing data rates, e.g., 60%.

4.2 Edgar Anderson's iris data

The next example is the famous Fisher's or Anderson's iris data set, which gives measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of three species of iris. The species are "Iris setosa," "versicolor," and "virginica" [1]. In R, "iris" is a data frame with 150 observations and 5 variables.

This data set was used as an example of discriminant analysis by Fisher, and it is a typical test case for many classification techniques in machine learning. Note that the data set only contains two clusters with a rather obvious separation.

One of the clusters contains Iris setosa, while the other cluster contains both Iris virginica and Iris versicolor and is not separable without the species information Fisher used.

The three methods were investigated within the same framework as in the previous experiment for spam/non-spam. Here, we shall pretend that iris spaces (5th variable) are not partially measured. Figure 2 shows the results of three runs.

Identical data sets corresponding to the missing rate were used to evaluate the three methods. Our method ("rfImput.smspvised") was the best of the three, irrespective of the missing data rate. The kNN method performed fairly well because of the few dimension of the iris data.

Whereas the experiment on the spam data involved alternatives, the one on iris data involved a threefold choice. Therefore, the decreases for the iris data (Fig. 2) are sharper than those of spam data (Fig. 1).

Typically, semi-supervised learning adds a large amount of unlabeled data to a small amount of labeled data for training. That is, the missing rate of the responses (y) should be large compared with that of the predictors (x). Hence, we set the missing rate of y to be 90% and performed the same

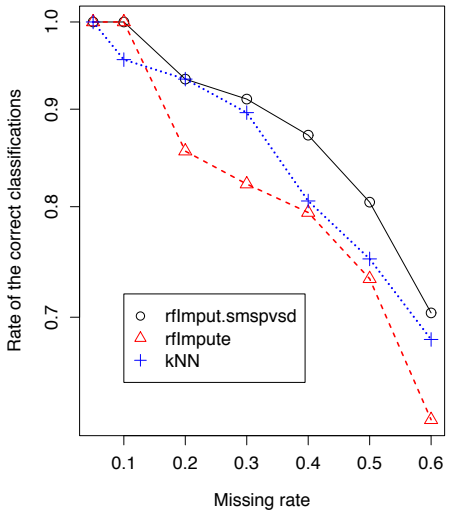

Figure 2: Correct classification for semi-supervised iris data

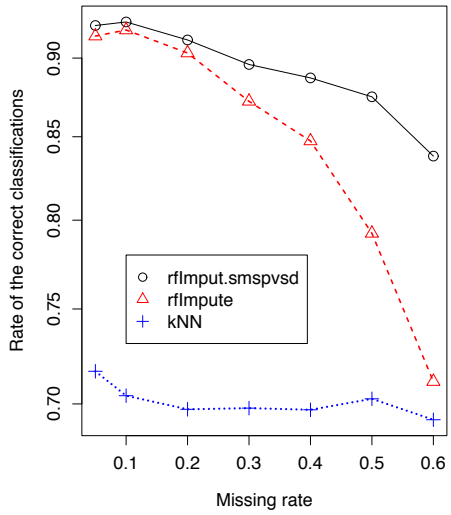

Figure 3: Correct classification after conducting semi-supervised learning on spam/nonspam data (90% missing for responses)

experiment as we did on the spam/non-spam data.

Figure 3 shows the results. Since the missing data structure depends on a seed of the randomization, the rate of correct classifications does not always decrease monotonously. Despite the lack of monotonicity, the relative relationship of the three is not different from Fig. 2; our method is superior to the other methods, and kNN is the worst. Note that the degradation in performance with our method is relatively small compared with the other methods.

The experiments were not performed using iris data because the sample size is too small, only 150, to apply semi-supervised learning.

5. CONCLUSIONS

In case of context-aware services, we set the response variables to the services to be provided, and the explanatory variables to the context information statistics. Usually, the explanatory variables contain some missing data. It is obvious that missing at random (MAR)[6], wherein the missing depends on only observations and not non-observations, is superior to missing completely at random (MCAR), wherein the missing does not depend on the variables in the assumed model. Random forests is subject to the MAR assumption, so it gives better results than those of conventional methods such as kNN.

Moreover, our method does not take account of the effects on data selection biases, because all cases or obtained data are used as they are. The situation or condition under which the complete data are obtained is often restricted. We hope that our method will be used by many researchers in the future.

6. ACKNOWLEDGMENTS

This work was supported by a Grant-in-Aid for Scientific Research No. 21300320 given to Tsunenori Ishioka.

7. REFERENCES

[1] R. A. Becker, J. M. Chambers and A. R. Wilks, The New S Language. *Wadsworth & Brooks/Cole*, 1988.

[2] L. Breiman, Random Forests, *Machine Learning*, **45** (1), 5–32, 2001.

[3] L. Breiman, *Manual for Setting Up, Using, and Understanding Random Forest V4.0*, `http://oz.berkeley.edu/users/breiman/Using_random_forests_v4.0.pdf`, 2003.

[4] L. Breiman and A. Cutler, Random Forests, `http://www.stat.berkeley.edu/~breiman/RandomForests/` updated March 3, 2004.

[5] CRAN, *Package randomForest*, `http://cran.r-project.org/web/packages/randomForest/randomForest.pdf`

[6] A. Gelman and J. Hill, *Data Analysis Using Regression and Multilevel/Hierarchical Models*, Cambridge University Press, 2007.

[7] k-Nearest Neighbour Classification, R Documentation, knn {class}, `http://stat.ethz.ch/R-manual/R-patched/library/class/html/knn.html`

[8] A. Liaw, Missing Value Imputations by randomForest, *R Documentation*, `http://www.stat.ucl.ac.be/ISdidactique/Rhelp/library/randomForest/html/rfImpute.html`

[9] C. L. Nicholas and F. O. Andrew, yaImpute: An R Package for kNN Imputation, *Journal of Statistical Software*, **23** (10), Jan 2008.

[10] The R Project for Statistical Computing, `http://www.r-project.org/`

[11] UCI Repository of Machine Learning Databases, `http://www.ics.uci.edu/~mlearn/MLRepository.html`