



Supervised knowledge discovery from incomplete data

A. Kalousis & M. Hilario
*Department of Computer Science,
University of Geneva, Switzerland.*

Abstract

Incomplete data can raise more or less serious problems in knowledge discovery systems depending on the quantity and pattern of missing values as well as the generalization method used. For instance, some methods are inherently resilient to missing values while others have built-in methods for coping with them. Still others require that none of the values are missing; for such methods, preliminary imputation of missing values is indispensable.

After a quick overview of current practice in the machine learning field, we explore the problem of missing values from a statistical perspective. In particular, we adopt the well-known distinction between three patterns of missing values—missing completely at random (MCAR), missing at random (MAR) and not missing at random (NMAR)—to focus a comparative study of eight learning algorithms from the point of view of their tolerance to incomplete data. Experiments on 47 datasets reveal a rough ranking from the most resilient (e.g., Naive Bayes) to the most sensitive (e.g., IB1 and—surprisingly—C50rules). More importantly, results show that for a given amount of missing values, their dispersion among the predictive variables is at least as important as the pattern of missingness.

1 Machine learning and the problem of missing values

Although missing values are quite common in machine learning applications such as medical diagnosis, relatively little research effort has been devoted to them. Often, the problem of incomplete data is evacuated hastily in the pursuit of other research interests. For instance, of the 23 classification algorithms studied in the Statlog project [6], barely half could tackle missing values whereas all the others required complete data. The policy adopted in this project was the simple expedient of replacing missing values of a predictor by its global mean or median.

Some learning methods are inherently tolerant to incomplete data and thus require no specialized mechanisms for handling missing values. For instance, certain techniques based on density estimation ignore missing values with no direct consequences. For instance, Naive Bayes simply skips over missing values

when computing distributions of attributes. In the general case, where the basic inductive approach requires complete data, implementations can vary depending on whether the designers have taken pains or not to integrate some mechanism for coping with missing values. When they haven't, the only possibility open to the user is to fill in all missing values before calling the learning algorithm. However, more and more learning methods exhibit embedded strategies for handling missing values, thus eliminating the need for preliminary missing value imputation.

In the simplest form of instance-based learning (e.g., IBL as we find it in the MLC++ library[7]) missing values are handled by means of a minor change in the distance metric (typically the Euclidean distance for continuous and the Hamming distance for discrete attributes). When two instances both have missing values on the same attribute, their distance on that feature is zero while the maximal distance is assigned when only one of them has a missing value.

Linear discriminants use a mean imputation strategy, i.e. replace missing values of an attribute with the mean of the attribute. Ripper incorporates a bias against missing values into the rule building process; any test on an attribute whose value is unknown returns a failure, so that the learner focuses on completely known features in selecting rule preconditions [3].

Unknown values are handled in C4.5 by means of several changes in the basic algorithm [8]. During training, the information gain measure of each attribute is weighted by the relative frequency of cases where its value is known; this penalizing attributes with missing values in the feature selection process. Once a test feature has been selected, a case is normally sent down the branch corresponding to its value for that feature. However, a case with a missing value is fractioned among the alternative branches proportionally to their respective probabilities (computed on the basis of known values). During the test phase, a query instance that has no value for a test attribute is also fractioned among the outgoing branches, which in effect output class distributions. Ltree [4] is a multivariate or oblique decision tree which follows basically the same strategy for handling missing values as C4.5.

2 A statistical perspective on missing values

Assume that the data set can be divided into an observed component X^o and a missing component X^m . We formalize the notion of a missing data mechanism by defining a missing data indicator matrix R such that $R_{ij} = 1$ if x_{ij} is observed and $R_{ij} = 0$ if x_{ij} is missing. Both the data generation process and the missing data mechanism are considered to be random processes, with joint probability distribution given by :

$$P(X, R | \theta, \phi) = P(X | \theta) P(R | X, \phi)$$

We use the parameters θ for the data generation process and a distinct set of parameters, ϕ , for the missing data mechanism.

We can distinguish three types of missing data mechanism [Schaffer 97][10]:

Missing completely at random (MCAR): The probability that x_{ij} is missing is independent of the values in the data vector X . Simple approaches like mean imputation or regression imputation can handle only data that is MCAR.

Missing at random (MAR): The probability that x_{ij} is missing is independent of the values of missing components of the data but may depend on the values of the observed components X^o . For example, x_{ij} may be missing for certain values of x_{ik} ($k \neq j$) provided that x_{ik} is always observed. Full maximum likelihood and Bayesian approaches can handle data that is MAR (and also MCAR).

Not missing at random (NMAR): The probability that x_{ij} is missing may depend on the value of x_{ij} . If it is a function of x_{ij} the data is said to be censored. For example, if a sensor fails when its input exceeds some range its output will be censored. No general approaches exist for NMAR data. When data are missing in a systematic way, the absence of information itself conveys information. Here prior domain knowledge or knowledge of the conditions in which data were collected should be used to decide on how to handle missing data.

3 Experimental Setup

The objective of this work was to investigate the behaviour of machine learning algorithms with respect to missing values. The independent variable was the predictive accuracy of the learners and the experimental parameters were the volume of missing values (expressed as the percentage of the total amount of attribute values), the number of attributes with missing values, and the pattern of missingness as defined in Section 2.

Eight classification algorithms were selected for the empirical study: two rule inducers (C5.0-rules [9] and Ripper [3]), one nearest neighbour method (IBL), [7], one orthogonal (C5.0-tree [9]) and one oblique decision tree inducer (Ltree, [4]), a Naïve Bayes inducer, [7] and a linear discriminant [4].

We selected 47 datasets with no missing values, most of them from the UCI repository [1]. Various levels of missing values were introduced ranging from 1% to 40%. Since the pattern of missing values was an important dimension of this study, three suites of experiments were created, two of them corresponding to MCAR and one to MAR patterns of missing values.

3.1 MCAR experimental suite

In the MCAR suite of experiments the strategy that we followed to create missing values was very simple. We randomly introduced missing values at the levels of 1%, 5%, 10%, 15%, 20%, 25%, 30% and 40%. Two suites were created. In the first one, which from now on we will call **MCAR uniform**, the missing values were evenly distributed across all attributes, so the percentages given above correspond to the probability that an attribute of an instance will have a missing value. In the second one which we will call **MCAR half**, missing values were created only on half of the attributes, but for purposes of



comparison, the percentage of missing values was doubled in each case in order to produce the same total amount of missing values as MCAR uniform.

3.2 MAR experimental suite

Simulating MAR missing values was more difficult. As mentioned in the previous section, the pattern of MAR values depends on the observed values. So in order to introduce missing values one should condition their generation upon the distribution of the observed values. Here also we wanted to keep the same levels of missing values. Since the patterns of missing values should depend on the observed values we chose to create attribute pairs (X,Y) , where Y is the attribute into which we introduced missing values and X is the attribute on the distribution of which the missing values of Y were conditioned ($P(Y=\text{mis}|X)$). So in this case we introduced missing values in only half of the attributes of each data set. Since we wanted to keep the percentage of missing values at the same level overall, we had to alter the percent of missing values of the individual attributes. Thus in the case of $k\%$ of missing values over the whole data set, each attribute on which we introduced missing values has a $2k\%$ of missing values. For each of the X attributes we estimated its $4k$ percentile, and for all instances that have the X attribute value in the $[\min\{X\}, 4k]$ interval we created a missing value for the Y attribute with probability 0.5. More formally $P(Y=\text{mis}|X < 4k) = 0.5$, $P(Y=\text{mis}|X > 4k) = 0.0$.

3.3 Evaluation procedure

For each experimental suite, we evaluated the performance of each learning algorithm using 10-fold cross-validation. The mean error was computed for each algorithm and level of missing values to give a rough indication of how predictive behaviour varies with respect to data incompleteness. To quantify more precisely the sensitivity of each algorithm to a given level of missing values, we computed the difference between the mean error at that level and at the control level (with no missing values). This measure gives a rough idea of an algorithm's resilience to missing values independently of the other algorithms. The findings presented in the next section were based on a total of 9071 experiments (47 data sets $\times 8$ learning algorithms $\times 8$ levels of missing values, including the control level $\times 3$ series $+ 47$ at the control level).

4 Findings

This section describes experimental results on the tolerance of learning algorithms to data that are missing completely at random (4.1 and 4.2) or missing at random (4.3).



4.1 MCAR UNIFORM

All algorithms' mean error, except for Naïve Bayes, increases seriously with the increasing percentage of missing values as it is obvious from the mean error results given in **figure 1**.

One of the most striking outcomes revealed by the mean error curves is the behaviour of the c50rules algorithm. Although it has a way to handle missing values it exhibits one of the worst performances. Its mean error increases from 13.79% in the control cases, to 36.30% at the 40% level of missing values. Actually c50rules has the second lowest mean error in the control cases, after c50boost, and at the 40% level it becomes the second highest just after the linear discriminant algorithm. On the other side of the spectrum there is Naïve Bayes, which exhibits the best scaling behaviour. Although its mean error at the base level is the highest (22.97%), it increases very gradually with missing values and attains only 26.08% and at the 40% level. This behaviour makes it the third best algorithm at the 40% level in terms of the mean error, even better than C50. Ltree also exhibits quite good scaling behaviour, rising from the fourth place at the control level to the first in the 40% level, together with C50boost.

To establish a clear picture of the scaling behaviour of each algorithm, **figure 1** depicts the difference between the mean error of each algorithm at a specific level of missing values and its mean error at the control level (i.e. no missing values). This figure exhibits in the clearest way the resilience of each algorithm to the specific pattern of missing values. It is quite obvious that Naïve Bayes is by far the most resilient algorithm. Its highest increase in error is always less

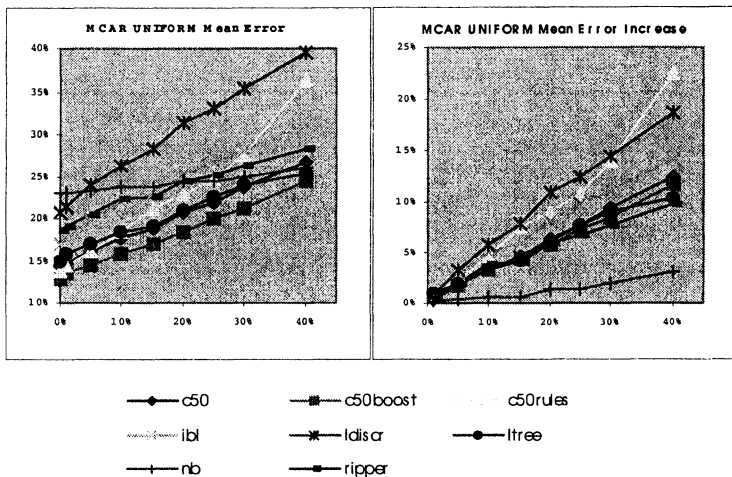


Figure 1: Mean error and mean error increase for each algorithm. The x-axis represents the percentage of missing values and the y-axis corresponds to each title.

than 0.04. In the next group we find Ripper, Ltree, C50boost and C50tree presenting similar behavior in terms of resilience. Next is IBL. In the group of the least resilient algorithms we find linear discriminants, which is expected since it uses the simple strategy of mean imputation, and strangely enough, C50rules.

4.2 MCAR HALF

In this suite of MCAR experiments the behaviour of the algorithms is different from the one we get in the uniform case. We no longer observe radical changes in the performance of the algorithms as it is the case in MCAR uniform suite. All algorithms are more resistant to that pattern of missing values and their error does not increase as much as in the suite of MCAR uniform.

Again C50rules exhibits one of the biggest increases in the mean error but now this increase is less than 10% at the 40% level of missing values. In terms of the mean error it drops from the second position in the control cases, to the fourth in the 40% missing values.

Naïve Bayes still exhibits the smallest increase in the mean error, but now since the rest of the algorithms are also resilient to the specific pattern of missing values, this does not result in any significant improvement to its ranking.

Ltree remains quite resilient to missing values and has one of the smallest increases in error, that gives it now the first position at the 40% level of missing values where it again overcomes C50boost. Ripper also exhibits one of the smallest increases in error but since it has a high error in the base cases its ranking does not improve significantly, although its performance in terms of the

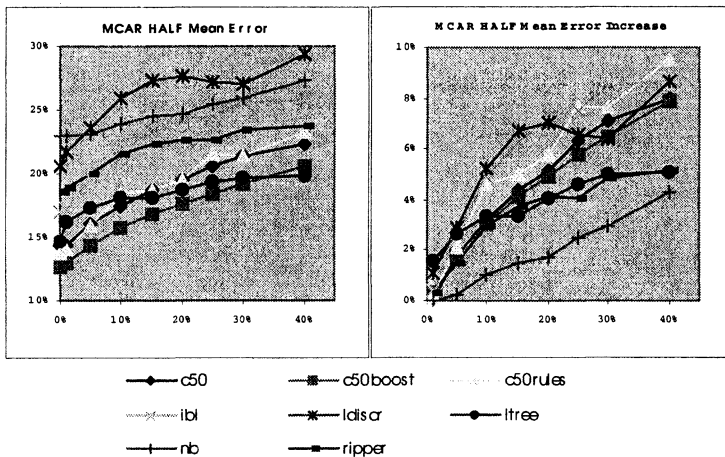


Figure 2: Mean error and mean error increase for each algorithm. The x-axis represents the percentage of missing values and the y-axis corresponds to each title.

mean error approaches that of C50rules and C50tree. As a matter of fact, given 40% missing values, the difference in mean error between Ripper and C50rules is not statistically significant at the 0.05 level using a sign test.

These findings suggest that for a given overall percentage of missing values, the decisive factor is their dispersion: more concretely, the same amount of missing values will have a stronger negative impact on learning if it were distributed over many attributes rather restricted to a small number of these.

4.3 MAR

In the MAR suite, the behaviour of the algorithms is similar to the one observed in the MCAR half suite, as it can be seen by examining the curves of mean errors and mean error increases (figure 3). C50rules again exhibits one of the highest error increases, together with IBL and Ldiscr. Naïve Bayes is again the one exhibiting the smallest increase in error, and then Ltree and Ripper follow. Ltree takes again the highest position at the 40% level. These observations confirm and extend the conclusion drawn in the Section 4.2; the dispersion of missing values is not only important, it also seems to have a stronger adverse effect on predictive accuracy than does the pattern of missing values. However, this result calls for a few caveats which are discussed in the next section.

5 Discussion

The predominance of dispersion over the pattern of missing becomes more pronounced as the percentage of missing value increases. At low levels of

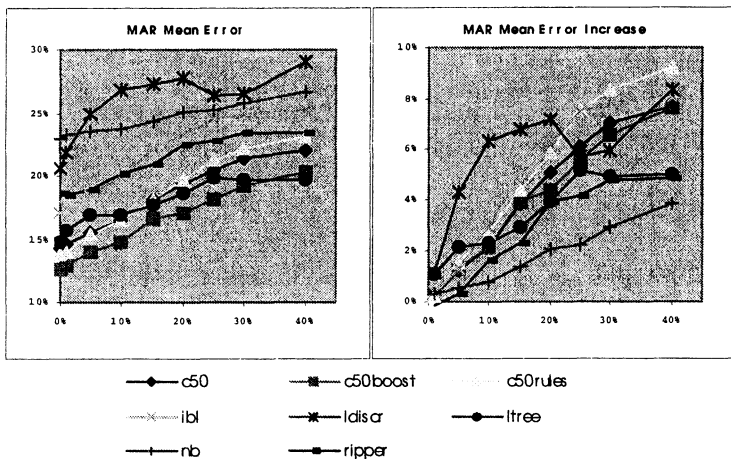


Figure 3: Mean error curves and mean error increase for each algorithm. The x-axis depicts the percentage of missing values and the y-axis corresponds to each title.

missing values ($\leq 20\%$) the mean errors of the algorithms are similar for the two MCAR suites and are slightly higher than for the MAR suite, except for Naïve Bayes and linear discriminants (**figure 4**). It seems that for low percentages of missing values the determinant factor of learning behavior is the actual pattern of missing values, (i.e. whether they are MCAR or MAR) and not how they are distributed among the attributes. More intriguingly, MCAR values entail more serious deterioration in predictive accuracy than MAR; this is somehow in conflict with statistical theory which considers MCAR cases easier to handle. In statistics the simplest tools are used for the MCAR, e.g. removal of instances with unknown values, or mean value imputation. Such tools are not appropriate for classification problems that appear in the knowledge discovery field, since their use assumes problems that can be approximated by multinormal distributions. But classification problems are typically multicluster problems and cannot be approximated by multinormal distributions, so that simple statistical techniques are not sufficient.

Moving to the higher levels of missing values the behaviour now changes and the determinant factor is how the missing values are distributed among attributes. All algorithms exhibit worse performance when the missing values are equally distributed among all attributes (MCAR uniform). Contrariwise they are much more resilient in the case that the same overall proportion of missing values is concentrated in a smaller number of attributes (MAR, MCAR half). The only notable exception to that pattern of behaviour is Naïve Bayes whose behaviour is in the limit better for the MCAR uniform suite. A possible explanation for the fact that algorithms exhibit a better performance when only half of the attributes are corrupted is that the information still available in the first uncorrupted half is sufficient for describing the concepts better than in the case where the missing values are equally distributed among all the attributes.

As we see in **figure 4**, the performance of the algorithms above a certain level of missing values ($\cong 20\%$) converges for the suites of MAR and MCAR half and is much better than that of MCAR uniform. This convergence can be explained as follows. When we have more than 20% missing values over the entire data set, this amounts to more than 40% of missing values on half the attributes. With such high levels of missing values the volume of missing values is dominating the problem and not the actual pattern of missing values.

When examining the resilience of the algorithms to missing values on the basis on mean error increase, we see that the most tolerant algorithm (i.e. the one presenting the smallest error increase) in all the different suites of experiments and levels is Naïve Bayes (**figures 1,2,3**). In one suite of experiments (MCAR uniform) its tolerance is so good that its ranking relative to the other algorithms is significantly improved at the highest level of missing values. The way Naïve Bayes handles missing values is quite appropriate for the MCAR uniform pattern, since Naïve Bayes computes distributions of attributes; those distributions do not change in that suite of experiments, and this is why it exhibits better performance in this suite of experiments contrary to all the other algorithms.



Following Naïve Bayes in tolerance are Ltree and Ripper. They exhibit similar performance in terms of mean error increase, with Ripper being slightly better. It seems that the bias of Ripper against attributes with missing values is quite appropriate. The difference from the rest of the algorithms is stronger in the cases of MCAR half and MAR. The tolerance of Ltree results in taking the first position at all the suites of experiments for the highest level of missing values, overcoming even C50boost which produces its predictions by combining multiple learned models.

A third cluster is composed of C50 and C50boost, whose similar performance is understandable since C50boost builds on C50. Finally Ldiscr, IBL and C50rules exhibit the highest sensitivity to missing values. This seems logical for IBL and Ldiscr whose built-in missing value mechanism is primitive. But in the case of C50rules the observed behaviour is counter intuitive. C50rules creates rules using trees produced by C50, thus we would expected it to exhibit a similar performance with C50, but it seems that the way the rules are extracted from the trees makes the algorithm more sensitive to missing values.

6 Summary and future work

In this paper the results of an experimental study were presented. The goal of the experimental study was to examine the resilience of classification algorithms to different patterns and different levels of missing values. To this end we introduced the notions of MAR and MCAR from the statistical community, and we artificially created missing values simulating those patterns.

The problem of missing values has been given relatively little attention in the field of machine learning, and as far as we know this is the first comparative study of classification algorithms with a focus on their capability to handle missing values. While this study confirms statistical insight on the importance of the pattern of missingness, it also reveals that the distribution of unknown values among the predictors plays an equally (if not more) decisive role, depending on the percentage of missing data. In a subsequent study we plan to examine whether the performance of classification algorithms can be improved by combining them with data pre-processing tools from the statistical community (e.g. multiple imputation of missing values [10]).

References

- [1] Blake, C., Keogh, E., Merz, C.J. *Repository of machine learning databases*. <http://www.ics.uci.edu/~mllearn/MLRepository.html>, University of California, Irvine, Dept. of Information and Computer Science 1998.
- [2] Breiman, L. *Classification and Regression Tress*, CRC press, 1983.
- [3] Cohen, W.W. Fast Effective Rule Induction. *Proc. 12th Int. Conf. on Machine Learning*, Lake Tahoe California, pp. 115-123, 1995.
- [4] Gama, J., Brazdil, P. Linear Tree, *Intelligent Data Analysis*, **3**, pp. 1-22, 1999.

- [5] Ghahramani, Z., Jordan M.I. *Learning from Incomplete Data*, Technical Report, MIT C.B.C.L. Paper No 108, 1994.
- [6] Michie, D., Spiegelhalter, D.J., Taylor, C.C. *Machine Learning, Neural and Statistical Classification*, Ellis Horwood, 1994.
- [7] Kohavi, R., Sommerfeld, D. & Dougherty, J. *Data mining using MLC++*. *Proc. of the Int. Conf. on Tools with AI*. 1996.
- [8] Quinlan, J.R. *c4.5 : Programms for Machine Learning*, Morgan Kaufman Publishers, 1992.
- [9] Rulequest. <http://www.rulequest.com>
- [10] Schaffer, J.L. *Analysis of Incomplete Multivariate Data*, CRC Press, 1997.

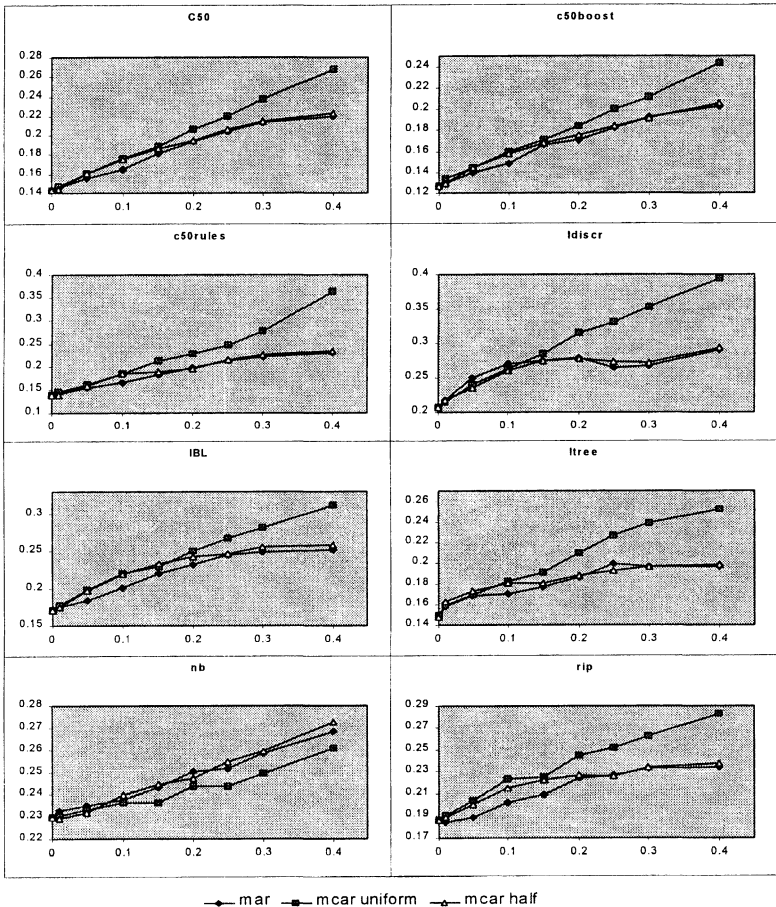


Figure 4: Comparing evolution of mean errors for each algorithm over the three suites.