Classification of Datasets with Missing Values: Two Level Approach

Ivan Bruha

McMaster University, Dept. Computing & Software, Hamilton, Ontario, L8S4K1, Canada

Abstract. One of the problems of pattern recognition (PR) are datasets with missing attribute values. Therefore, PR algorithms should comprise some routines for processing these missing values.
There exist several such routines for each PR paradigm. Quite a few experiments have revealed that each dataset has more or less its own 'favourite' routine for processing missing attribute values. In this paper, we use the machine learning algorithm CN4, a large extension of well-known CN2, which contains six routines for missing attribute values processing. Our system runs these routines independently (at the base level), and afterwards, a meta-combiner (at the second level) is used to generate a meta-classifier that makes up the overall decision about the class of input objects.
This knowledge combination algorithm splits a training set to *S* subsets for the training purposes. The parameter *S* (called 'foldness') is the crucial one in the process of meta-learning. The paper focuses on its optimal value. Therefore, the routines used here for the missing attribute values processing are only the vehicles (for the function of the base classifiers); in fact, any PR algorithm for base classifiers could be used. In other words, the paper does not compare various missing attribute processing techniques, but its target is the parameter *S*.

1 Introduction

There exist several aspects of processing real-world databases by pattern recognition (PR) algorithms. These PR algorithms have to take into account various aspects of imperfection of existing data collected within the real applications. One of these aspects are missing (unknown) attribute values in such real-world databases. Therefore, robust PR algorithms have to exhibit some routines for processing these missing attribute values when acquiring knowledge from real-world databases.

The aspect of processing missing values has been discussed and analyzed by several researchers in the field of machine learning [1], [4], [6], [10], [11], [12], [13], [14]. Our paper [6] discusses both the sources of 'missingness' and the six routines for processing missing attribute values; the rule-inducing machine algorithm CN4, a large extension of the well-known algorithm CN2 [3], [7], [8] has been used for the experimental analysis. The paper [6] concludes that each dataset needs more or less its own 'favourite' routine for processing missing attribute values. It evidently depends on the magnitude of noise and source of missingness in each dataset. The conclusion is obvious: All the routines should be independently run on a small subset (window)

Bruha I.
Classification of Datasets with Missing Values: Two Level Approach.
DOI: 10.5220/0003017800900098
In *Proceedings of the 10th International Workshop on Pattern Recognition in Information Systems (ICEIS 2010)*, page
ISBN: 978-989-8425-14-0

of a database supplied and a suitable routine should be selected according to their classification accuracies.

This paper describes another way of processing missing attribute values. We were inspired by the idea of multiple knowledge, multi-strategy learning, and meta-learning, particularly by the concept of combiner and stack generalizer [9]. This concept is employed as follows. The algorithm CN4 processes a given database for each of six routines for missing attribute values independently. We can thus view the CN4 algorithm with various routines as independent *base learners*. Consequently, we obtain (at the base level) six independent *base classifiers*. Also, a *meta-database* is derived from the results of the base classifiers, and then a *meta-learner* induces a *meta-classifier*. We call the entire system *meta-combiner* (namely *Meta-CN4* to emphasize the origin of the algorithm).

Hence, if an unseen object is to be classified, then each base classifier yields its decision (class of the input unseen object), and the meta-classifier combines their results in order to produce the final (over-all) decision about the class of the given input object.

In this paper, we focus on one important parameter of the meta-combiner. If we go to its detailed structure, it encompasses several parameters that are to be set up by the designer or user of the system. One of the crucial parameters is the number S of subsets which a training set is partitioned into during the meta-learning. We are then talking about S-fold meta-combiner. Usually, the 'foldness' S is equal to 2 or the size of the training set. This paper exhibits the performance of the meta-combiner for processing missing attribute values as a function of S . The results of experiments for various values of the parameter S and various percentages of missing attribute values on real-world data are presented and analyzed.

Note that the routines used here for the missing attribute values processing are only the vehicles (for the function of the base classifiers); in fact, any PR algorithm for base classifiers could be used. In other words, the paper does not compare various missing attribute processing techniques, but its target is the parameter S .

The way CN4 processes missing attribute values are briefly presented in Section 2. Section 3 introduces the principle of the meta-combiner. Experiments exhibiting the performance of Meta-CN4 are discussed in Section 4. The results are analyzed in Section 5.

2 Processing of Missing Attribute Values

The inductive rule-generating algorithm CN4 generates decision rules from a set of K training examples (objects), each accompanied by its desired class C_r , r = 1,...,R . Examples (objects) are formally represented by N attributes which are either discrete (symbolic) or numerical (continuous). A discrete attribute A_n comprises J(n) distinct values $V_1,...,V_{J(n)}$; a numerical attribute may attain any value from a continuous interval. Numerical attributes are usually converted to a sequence of intervals by a discretization algorithm, see e.g. [2].

To deal with a real-world situation, it is necessary to process incomplete, imperfect data, i.e. data with missing attribute values. Six routines for processing of missing attribute values were designed for CN4 [6].

The following natural ways of dealing with missing attribute values were incorporated:

(i) ignore an example (object) with missing values (routine *Ignore*),
(ii) consider the missing value as an additional regular value for a given attribute (routine *Missing*), or
(iii) substitute the missing value for matching purposes by a suitable value which is either
- the most common value (routine *Common*), or
- a proportional fraction (routine *Fraction*), or
- a random value from the probabilistic distribution (routine *Random*), or
- any value of the known values of the attribute that occur in the training set (routine *Anyvalue*).

Treating missing attribute values is determined by the following statistical parameters (here the classes are subject to the index r=1,...,R , attributes A_n for n=1,...,N , their values j=1,...,J(n)):

- the *over-all absolute* frequencies $F_{n,j}$ that express the number of examples exhibiting the value V_j for each attribute A_n ;
- the *class-sensitive absolute* frequencies $F_{r,n,j}$ that express the number of examples of the class C_r exhibiting the value V_j for each attribute A_n ;
- the *over-all relative* frequencies $f_{n,j}$ of all known values V_j for each attribute A_n ;
- the *class-sensitive relative* frequencies $f_{r,n,j}$ of all known values V_j for each attribute A_n and for a given class C_r .

The underlying idea for learning relies on the class distribution; i.e., the *class-sensitive* frequencies are utilized. As soon as we substitute a missing value by a suitable one, we take the desired class of the example into consideration in order not to decrease the noise in the data set. On the other hand, the *over-all* frequencies are applied within classification.

(1) Routine Ignore: Ignore Missing Values

This strategy simply ignores examples with at least one missing attribute value before learning. Consequently, this approach does not contribute to any enhancement of processing noisy or partly specified data.

(2) Routine Missing: Missing Value as a Regular One

A missing value is considered as an additional attribute value. Hence, the number of values is increased by one for each attribute that depicts a missing value in the training set.

(3) Routine Common: The Most Common Value

This routine needs the class-sensitive absolute frequencies $F_{r,n,j}$ to be known before learning and the over-all frequencies $F_{n,j}$ before classification. A missing value of a discrete attribute A_n of an example belonging to the class C_r is replaced by the *class-sensitive common* value which maximizes the Laplacian formula $\frac{F_{r,n,j}+1}{F_{n,j}+R}$ over j for the given r and n. A missing value within classification is replaced by the *over-all*

common value which maximizes $F_{n,j}$ over subscript j . We use here the Laplacian formula within learning because it prefers those attribute values that are more predictive for a given class in the contrary to the conventional 'maximum frequency' scheme.

(4) Routine Fraction: Split into Proportional Fractions

The learning phase requires that the relative frequencies $f_{r,n,j}$ above the entire training set are known. Each example **x** of class C_r with a missing value of a discrete attribute A_n is substituted by a collection of examples before the actual learning phase as follows: missing value of A_n is replaced by all known values V_j of A_n and C_r. The weight of each split example (with the value V_j) is

$$w_j = w(\mathbf{x}) * f_{r,n,j} , \quad j=1,...,J(n)$$

where $w(\mathbf{x})$ is the weight of the original example **x** . The original weights $w(\mathbf{x})$ are provided by the designer of the training database of a given task; usually (by default) it is 1 .
If a training example involves more missing attribute values, then the above splitting is done for each missing value.

(5) Random Value: Generate Attribute Value Randomly

A missing attribute value is replaced by one of the values of the given attribute by utilizing a random number generator; it yields a random number in the range <0; 1> which is exploited to the select corresponding value by utilizing the distribution of its attribute values. In the learning phase, the distribution is formed by the class-sensitive relative frequencies $f_{r,n,j}$ of all known values V_j for each attribute A_n and for a given class C_r . In the classification phase, the over-all relative frequencies $f_{n,j}$ are used.

(6) Routine Anyvalue: Any Value Matches

A missing value matches any existing attribute value of an example (object), both in learning and classification. This routine in fact emulates the situation that a designer of a training database does not care about a value of a certain attribute for a given example (so-called *dont-care* scenario).

3 Methodology: Meta-combiner

As we stated at the beginning, each database has its own ‘favourite’ routine for processing of missing attribute values. We have exploited the rule-inducing machine learning algorithm CN4. It contains six routines for missing values processing. Our system runs these routines independently, and afterwards, a meta-learner is used to derive a meta-classifier that makes up the overall (final) decision about the class of input unseen objects.

Each of the six *base learners* (CN4 with different routines for processing missing attribute values) generates a *base classifier*. Afterwards, the decisions of the base classifiers form a *meta-database,* a set for training *meta-objects* (examples) for the *meta-learner*. The meta-learner then generates a *meta-classifier*. The meta-classifier does not select the best base classifier (routine for processing missing attribute values)

but rather combines the decisions (predictions, classes) of all the base classifiers. In the classification phase, the base classifiers first derive their predictions (classes, decisions); then a meta-object is deduced from these predictions which is then classified by the meta-classifier.

More precisely, the *meta-combiner* consists of two phases: meta-learning and meta-classifying; we will now define both phases in detail.

For the meta-learning purposes, a training set is split into two subsets: the *genuine-training* and *examining* ones. The genuine-training subset is applied for inducing the base classifiers; the examining one for generating a meta-database.

Let q be the q-th base classifier, q=1...,Q (where Q is the number of the base classifiers; in our project Q=6). Each *examining* example **x** of the examining subset generates a *meta-object* of the meta-database as follows. Let z_q be the decision (class) of the q-th base classifier for the examining object **x** ; then the corresponding meta-object of the meta-database looks as follows:

$$[z_1, \ldots , z_Q , Z]$$

where z_q , q=1...,Q is the decision the of q-th base classifier, Z is the desired class of the input examining object.

Let T be a training set of K training examples, S be an integer in the range <2; K> . Let us assume to have Q different base learners BL_q , q=1,...,Q, and a meta-learner *ML*. The flow chart of the meta-learner looks as follows:

procedure Meta-Learning-Phase(T, S)

1. Partition the training set T randomly into S disjoint subsets of equal size (as equal as possible); let T_s be the s-th such subset, s=1,...,S, card(T_s) the number of its objects (examples); the splitting (partition) procedure has to preserve the original distribution of classes as in T.
2. Form S pairs [T_s , $T \setminus T_s$] , s=1...,S ; for each s , $T \setminus T_s$ is the genuine-training subset and T_s the examining one, generated from the training set T
3. Let *MetaDatabase* be empty
4. **for** s=1,...,S **do**
   - 4.1 Train all base learners BL_q using the genuine-training subset $T \setminus T_s$; the result is Q base classifiers BCl_q , q=1,...,Q
   - 4.2 Classify the examining objects from T_s by these base classifiers
   - 4.3 Generate card(T_s) meta-objects and add them to *MetaDatabase*

   enddo
5. Train the meta-learner *ML* using the meta-database *MetaDatabase* ; the result is a meta-classifier *MCl**
6. Generate the base classifiers BCl_q^*, q=1,...,Q using the entire training set T which will be used in the meta-classification

Similar scenario is applied for classifying an unseen object **x** :
procedure Meta-Classifying-Phase(**x**)

1. Classify the unseen object **x** by all Q base classifiers BCl_q^* (generated in the step 6 of the meta-learning phase); let the output of the q-th base classifier BCl_q^* be z_q , q=1,...,Q
2. Generate the corresponding meta-object $[z_1, ..., z_Q]$
3. Classify the above meta-object by the meta-classifier Mcl^*; its result (decision) is the class to which the given input object **x** is classified

The number S of split subsets is crucial for this system. Therefore, we call it *S-fold meta-learner* or *S-fold meta-combiner*. The paper [9] introduces two architectures of their meta-system: combiner and stacked generalization. Their combiner corresponds to the 2-fold and stacked generalized to the *K*-fold meta-learner.

This was the reason, why we have focused on the 'foldness' S of the meta-combiner. In the following, we present a few experiments whose purpose is to compare the performance of the *S*-fold meta-combiner for various values of the parameter S .

4 Experiments

The 'foldness' S is one of the crucial parameters of a meta-combiner. In order to find out empirically the dependance of the meta-combiner's performance on various values of its 'foldness' S, we carried out several experiments. Unlike the previous experiments and analysis [5] (where we compared Meta-CN4 and Meta-ID3 with the original CN4 with various routines for missing attribute value processing, as well as C4.5), here we have focused just on Meta-CN4; we have studied its performance for various values S and various percentages of missing attribute values on real-world data.

All the above experiments were tested on four databases that can be found in UCI machine learning depository (http://www.ics.uci.edu/~mlearn) and STATLOG project (http://www.the-data-mine.com/bin/view/Misc/StatlogDatasets):

- ***ThyroidGland*:** This task of diagnosis of thyroid gland disease has been provided by the Institute of Nuclear Medicine of Inselspital, Bern, Switzerland. The database has been used at the Dept. of Advanced Mathematics, University of Bern, and also in the project CN4. The entire set involves 269 patients' data. Each patient is described by 19 attributes, 5 of them are numerical attributes, the rest are symbolic ones; the average number of values per symbolic attribute is 4.9 . About 30% of attribute values are unknown. The task involves two classes; the frequency of the majority class is 72%.
- ***BreastTumor*:** This dataset has been provided by the Jozef Stefan Institute, the research group of Prof. Dr. I. Bratko, Ljubljana. It involves 288 examples and two classes. Each attribute is represented by 10 attributes, 4 of them are numerical; symbolic attributes exhibit on average 2.7 values per attribute. 0.7% of attribute values are unknown. The majority class has frequency 80%.
- ***Onco*:** The oncological data were used for testing in the Czech Academy of Sciences, Prague, Czechland, and also in the project CN4 [6]. The entire set involves

127 examples. Each example is represented by 8 attributes; 7 of them are numerical attributes, the only symbolic one involves 3 values. All attribute values are known. The task involves three classes; the frequency of the majority class is 50%.

- ***Soybean*:** This well-known data has been used in many various experiments in machine learning, namely within the AQ family. The set available involves 290 training examples and 15 classes. Each example is characterized by 24 attributes, all are symbolic ones with average 2.9 values per attribute. The set exhibits 3.7% unknown attribute values. There are 4 classes exposing the maximum frequency (14%).

Each database has been randomly split to two sets (70% training, 30% testing) and this scenario has been executed 10 times for each combination. The following table thus involves in each slot an average of classification accuracy (of testing sets) acquired from 40 runs, i.e. an average above all four databases.

We have to realize that the above splitting procedure has nothing common with the splitting procedure within the meta-learner. The 70% of training examples are furthermore split within the *S*-fold meta-learner into a genuine-training subset of the size 70*(*S*-1)/*S* % and an examining subset of the size 70/*S* % of the original database.

Following [9], we selected 2-fold, then the promising 4-fold meta-combiner (*S*=4), and also *S*=8, *S*=16, and *S*=32 for comparison.

To achieve extensive and comprehensive comparison of the above meta-combiners' behaviour we have also decided to find how classification accuracy depends on various percentage of missing attribute values in databases. Only one database (ThyroidGland) exhibits a reasonable size of 'missingness'. As for the remaining databases, to emulate various number of missing values, we have run the original data through a filter which randomly changes attribute values to missing ones. The filter procedure ('missingizer') has the percentage of missing attribute values as its parameter.

Table 1 comprises the average classification accuracy (in %) above all four databases for various percentage of missing values and various values of the parameter *S*.

Table 1. Average classification accuracy (in %) of Meta-CN4 as a function of unknownness and 'foldness' *S* (above all four databases).

	S=2	*S*=4	*S*=8	*S*=16	*S*=32
5%	79.6	80.6	80.7	80.7	79.6
10%	78.5	79.3	79.2	79.4	78
20%	77	77	77.4	77	74.7
30%	75.6	76.7	76.3	76	75.5

5 Conclusions

The two-level (or meta-level) approach to machine learning has been studied for many years. This research paper continues in our larger project whose purpose is to design, implement, and empirically compare the meta-learner *Meta-CN4* with other algorithms for processing of missing attribute values. Namely, this paper exhibits a portion of the above project that considers the importance of the 'foldness' *S* of such a meta-combiner as its crucial parameter. The only, but widely used criterion in our experiments was the classification accuracy acquired from testing sets.

By analyzing the results of our experiments we came to the following:

Although there were carried out the experiments only for a few values of the parameter *S*, we can observe that there is the 'optimal' value *S* that maximizes the classification accuracy. One can easily observe it namely along the series for *S*=32 that exhibits always worse performance than that for other values.

To be more precise, the statistical results of the t-test (with the confidence level 0.05) depict that the performance of the meta-combiner for *S*=4, *S*=8, and *S*=16 are statistically equivalent, but they are significantly better than that for *S*=2 and *S*=32.

Because of time limitations, we did not perform more experiments. We did not use the stack generalizer (fold *S*=*K*) because it is much more time consuming; the paper [9] indicates that the timing cost for the stack generalizer is much more larger than that for the meta-combiner for relatively small parameters *S*.

For the future research, we plan to perform more experiments and to study how the optimal value of the parameter *S* depends on a processed database. It is just our impression that even for this issue (to find an optimal value of *S*), we would need to introduce another 'meta-level'.

References

1. Batista, G., Monard, M.C.: An analysis of four missing data treatment methods for supervised learning. Applied Artificial Intelligence, 17 (2003), 519-533
2. Berka, P. and Bruha, I.: Various discretizing procedures of numerical attributes: Machine Learning, and Knowledge Discovery in Databases, Heraklion, Crete (1995), 136-141
3 Boswell, R.: Manual for CN2, version 4.1. Turing Institute, Techn. Rept. P-2145/Rab/4/1.3 (1990)
4 Bruha, I.: Unknown attribute values processing utilizing expert knowledge on attribute hierarchy. 8th European Conference on Machine Learning, Workshop Statistics, Machine Learning, and Knowledge Discovery in Databases, Heraklion, Crete (1995), 130-135
5 Bruha, I.: Unknown attribute values processing by meta-learner. International Symposium on Methodologies for Intelligent Systems (ISMIS-2002), Lyon, France (2002)
6 Bruha, I. and Franek, F.: Comparison of various routines for unknown attribute value processing: Covering paradigm. International Journal Pattern Recognition and Artificial Intelligence, 10, 8 (1996), 939-955
7 Clark, P. and Boswell, R.: Rule induction with CN2: Some recent improvements. EWSL'91, Porto (1991), 151-163
8 Clark. P. and Niblett, T.: The CN2 induction algorithm. Machine Learning, 3 (1989), 261-283
9 Fan, D.W., , Chan, P.K., Stolfo, S.J.: A comparative evaluation of combiner and

stacked generalization. Workshop Integrating Multiple Learning Models, AAAI, Portland (1996)
10. Fortes, I. et al.: Inductive learning models with missing values. Mathematical and Computer Modelling, 44 (2006), 790-806
11 Quinlan, J.R.: Induction of decision trees. Machine Learning, 1 (1986), 81-106
12 Quinlan, J.R.: Unknown attribute values in ID3. International Conference ML (1989), 164-8
13 Wu, X., Barbara, D.: Constraints in data mining of contents. ACM SIGKDD Explorations Newsletter (2002), 1931-1945
14 Zhang, S. et al.:'Missing is useful': Missing values in cost-sensitive decision trees. IEEE Trans. Knowledge and Data Engineering, 17, 12 (2005), 1689-1693