A Comparison of Strategies for Handling Missing Values in Rule Learning

Technical Report TUD–KE–2009-03

Lars Wohlrab, Johannes Fürnkranz
Knowledge Engineering Group, Technische Universität Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Knowledge
Engineering

Abstract

In this paper, we review possible strategies for handling missing values in separate-and-conquer rule learning algorithms, and compare them experimentally on a large number of datasets. In particular through a careful study with data with controlled levels of missing values we get additional insights on the strategies' different biases w.r.t. attributes with missing values. Somewhat surprisingly, a strategy that implements a strong bias against the use of attributes with missing values, exhibits the best average performance on 24 datasets from the UCI repository.

Contents

1 Introduction

In practical applications, some values in a database may not be known, and learning algorithms have to be able to deal with such unknown values and missing information. In this paper, we empirically compare several such approaches in a conventional separate-and-conquer rule learning algorithm. While there has been some work on how to deal with missing values in other areas, such as decision-tree induction, there has not been that much published work in this area in inductive rule learning. Even though most algorithms, such as CN2 (Clark and Niblett, 1989) or RIPPER (Cohen, 1995) have some way of dealing with missing values, their procedures are not described well. Most notably there is hardly any work that empirically compares different approaches.

To our knowledge, the only previous publication in this area is by Bruha and Franek (1996), who empirically compared five different strategies on four different strategies. Our experimental work, reported in this paper, significantly extends their study in various ways: First, we evaluate a larger number of strategies (eight vs. five) on a much larger set of databases (24 datasets with missing values plus three for which we systematically varied the fraction of missing values vs. originally four datasets). Second, we provide a deeper analysis of the results in terms of accuracy, fraction of selected tests with missing attributes, model size and run-time. Finally, we also study the performance with two different rule learning heuristics that implement quite different biases.

We will first briefly review the problem of unknown values (Section 2) and discuss common strategies for addressing it (Section 3). We then describe our experimental setup in Section 4, and discuss our results on semi-real and real-world datasets in Section 5. A discussion of related work can be found in Section 6 before we conclud in Section 7.

2 Unknown Values

There are several types of missing information, which may be dealt with in different ways. One can distinguish at least three different types of unknown attribute values (let **A** be an attribute, and **x** be an example for which the attribute value is missing):

- *missing value:* **x** should have a value for **A**, but it is not available (e.g., because it has not been measured).
- *not applicable value:* the value of **A** cannot be measured for **x** (e.g., the value of the attribute pregnant for male patients)
- *don't care value:* attribute **A** could assume any value for **x** without changing its classification

However, in practical applications one typically does not know which type of unknown value is the case. Most rule learning systems therefore do not discriminate between the different types and treat all unknown values uniformly. A notable exception is CN2, which can discriminate between *don't care* and *missing* values and employs different strategies for dealing with them.

Furthermore, unknown values have to be dealt with in two different phases, first when the rules are learned, and second when the rules are put to use. The difference between these phases lies in the amount of available training information: in the learning phase, one can try to use the class label of an example as additional information in order to, e.g., guess a correct value, which may improve the quality of the training examples. However, at classification time, this information is not available, and an unsupervised strategy, which does not use the class labels, has to be employed.

3 Strategies for Handling Unknown Values

The following subsections are used to introduce the particular strategies considered in this paper. Essentially, we can distinguish three principal approaches to handling unknown values:

- ignore examples with unknown values (*Delete* strategy)
- treat unknown values uniformly for all examples (the *Ignored Value*, *Any Value*, *Special Value*, and *Common Value* strategies)
- treatment of unknown values depends on the example (the *Pessimistic value*, *Predicted value*, and *Distributed value* strategies)

In the following, we will briefly review these approaches.

3.1 Delete Strategy

The simplest strategy is to completely ignore examples with unknown values. This does not require any changes in the learning algorithm, and the changes in the feature generation algorithm reduce to adding a simple filter that blocks examples with unknown values. The key disadvantage of this method is, of course, the waste of training data.

As examples usually cannot be discarded at classification time, a different treatment has to be applied there. The simplest solution is not to cover unknown values by any condition (the *Ignore* strategy, see below).

3.2 Ignored Value Strategy

The *Ignore* strategy simply ignores attribute values with unknown values, i.e., they cannot be covered by any feature. Thus, every feature that tests for the unknown value will be assigned the truth value false. Like the *Delete* strategy, this strategy can be easily realized in the feature generation phase, but is less wasteful with training data. In fact, one can say that it effectively exploits the entire known information. Note that this strategy is not applicable in decision tree induction, where typically each outcome of test is associated with a value. It is, e.g., realized in the RIPPER rule learning system (Cohen, 1995).

3.3 Any Value Strategy

This strategy does the opposite of the *Ignore* strategy: while the latter treats all missing values as uncovered, this strategy treats them all as covered. Thus it essentially corresponds to interpreting unknown values as *don't care* values.

3.4 Special Value Strategy

Another straight-forward approach is to treat an unknown value as a separate value for this attribute. This corresponds to including a separate feature that indicates whether the value of the attribute is unknown or known. This may be particularly adequate when we interpret unknown values as *not applicable* values.

In some sense, the *Special Value* strategy may be viewed as an extension of the *Ignore* strategy. While both strategies leave all examples with missing values uncovered by conventional tests, *Special Value* includes a separate feature that allows to cover these examples. On the other hand, it should be noted that in scenarios where attribute values are missing at random (and thus unknown values provide no information about the target-concept) this strategy has no advantage compared to *Ignore* as the best case would then be to learn no "special" conditions at all.

3.5 Common Value Strategy

All previous approaches do not look at the data when deciding on how to treat an unknown value. If we assume the existence of a "true" value for each unknown one, we can try to estimate this true value based on the known information.

The simplest approach for utilizing data-dependent information is to replace unknown values of discrete attributes by the most common value (the *mode* value), and unknown values of continuous attributes by their average value (the *mean* value). This method is quite straight-forward to realize, but it has its drawbacks, especially when the number of unknown attribute values is high.

3.6 Pessimistic Value Strategy

The *Pessimistic Value* strategy was recently introduced in (Gamberger et al., 2008; Lavrač et al., To appear). Its key idea is to hinder the utilisation of attributes with many unknown values by penalising the evaluation-heuristic for tests on unknown values. Thus, when evaluating candidate rules in the learning phase, tests on missing values are always counted as an error, i.e. positive examples with a missing value are not covered by any test based on this attribute, and, conversely, negative examples are covered by all tests based on this attribute. As every reasonable heuristic will penalise wrong classifications, this effectively decreases the estimated quality of conditions using attributes with missing values (and thus of rules that use such conditions).

Note that *Pessimistic Value* may be viewed as a mixture between the *Ignored Value* and the *Any Value* strategy, employing the former for positive and the latter for negative examples. It may also be seen as an adaptation of the "reduced information gain" approach for decision trees (Quinlan, 1989).

In contrast to the learning-phase, no class information is available when classifying new examples. Hence it is not possible to tie the decision about the coverage of an example to its class label. Thus, the *Ignored Value* strategy is used at classification time.

3.7 Predicted Value Strategy

Always guessing the mean or mode values for the missing value, as in the *Common Value* strategy, has the obvious disadvantage that it is not sensitive to the example in question. One can, instead, train a separate classifier for each attribute, which can then be used for replacing the missing value with an educated guess. In principle, any classifier can be used for this task, but lazy classifiers are particularly popular because they do not need to be trained unless a value is needed.

Therefore, the *Predicted Value* strategy considered here utilises a *nearest neighbor* classifier for that purpose. This corresponds to the assumption that "similar" examples are more suitable for estimating the true values of missing attributes. Consequently, instead of taking all available examples into account, as *Common Value* does, only the k most similar examples are used to determine the "common" value. Obviously, the validity of the basic assumption vastly depends on the proper choice of the similarity function. In this paper, all experiments were performed using `LinearNNSearch` from the Weka-library as similarity-function. Besides, the strategy can be parameterized by the number k of neighbours to be considered. Different variants of the *Predicted* strategy employing different values for k were examined in this paper. In order to differentiate the respective variants, the respective value of k is appended to the strategy's identifier, e.g. *Predicted.3* for $k=3$.

The NN-based *Predicted Value* strategy can also be seen as a generalisation of the *Common Value* strategy as it contains *Common* as special case for $k=\infty$ (or k equals the size of the dataset, respectively).

It may happen that all k nearest neighbours also have unknown values for the desired attribute. In this case, a fallback strategy should be used which is capable of handling missing values directly. Here, the *Ignore* strategy has been applied for that purpose.

3.8 Distributed Value Strategy

Finally, one can predict a probability distribution over all possible values. The example with the missing value is then divided into fractions. Each of these fractions receives a different feature-based representation, and an example weight $w_x \in [0,1]$, which indicates the degree to which this example can be covered (conventional, unweighted examples only have values 0 or 1). Implementing this approach requires a non-trivial modification of the rule learner that adapts all counting routines to weighted examples.

A variant of this technique is, e.g., used in the CN2 rule learner (Clark and Niblett, 1989; Clark and Boswell, 1991), which, in the learning phase, uses Laplace-corrected example weights, but for computing rule weights and for classifying new examples, uses equal fractions for all possible values of the missing attribute.

As the splitting of examples can lead to an enormous inflation of the dataset, it is advisable to limit this effect. Here a configurable weight-threshold has been applied, i.e. example-fractions whose weights fall below the given threshold after a split-operation are dropped. If all fractions have to be dropped, the *Ignore* strategy is used as a fallback strategy

again. In the experiments reported in this paper, we used seven weight thresholds ranging from 0.01 to 0.95. In order to label the respective variants, the fractional part of the threshold is appended to the strategy's identifier.

4 Experimental Setup

For our experiments, we implemented all eight strategies for handling missing values in a simple separate-and-conquer (SeCo) rule-learner with a top-down hill-climbing search for individual rules. Rules are greedily refined until no more negative examples are covered, and the best rule encountered in this refinement process (not necessarily the last rule) is returned. As a search heuristic, we used Laplace and the m-estimate in the setting recommended by Janssen and Fürnkranz (2008). We did not employ explicit stopping criteria or pruning techniques for overfitting avoidance. The resulting algorithm is essentially equivalent to CN2 (Clark and Niblett, 1989).

We compared these strategies on a number of datasets, both real and artificially prepared. In the UCI repository for machine learning databases (Hettich et al., 1998), we found 24 datasets[1] with missing values.

In addition, we selected three larger datasets without missing values (CREDIT-G, KRKP, SEGMENT) from which we removed varying fractions of the values of certain attributes in order to simulate different scenarios. In analogy to the term "value imputation", which denotes the replacement of a missing value with an educated guess, we call the opposite procedure of replacing a true value with an unknown value "*value amputation*". The artificial missing values were introduced according to the following two-step strategy:

1. *attribute selection*: In order to ensure that the selected attributes have a significant impact on the result, we selected the three most relevant attributes according to a χ^2-test.

2. *value removal*: From each of these three attributes, we removed m% of the values from the dataset, where the *amputation level* m was varied in steps of 15%.

By this means the original dataset is transformed into a family of six "amputed datasets" with 15% to 90% missing values for each amputed attribute.[2] Note that the second step, value removal, assumes that the values are missing at random. It is questionable whether this assumption holds in practice, but as we complement these results with results on 24 real-world datasets, we think this assumption is not crucial.

The two configurable strategies *Distributed* and *Predicted* have by default been configured with a minimum weight threshold of 0.05 (*Distributed.05*) and a neighbourhood of size 9 (*Predicted.9*) respectively, but we will also report results that show the effect of different parameter choices.

All methods were evaluated with 10-fold cross-validation within the Weka data mining library (Witten and Frank, 2005). In addition to the estimated predictive accuracy, the times required to learn and evaluate the respective models have been monitored as well. A more detailed description of the procedure and more detailed experimental results can be found in (Wohlrab, 2009).

[1] *audiology, auto-mpg, autos, breast-cancer, breast-w, breast-w-d, bridges2, cleveland-heart-disease, colic, colic-orig, credit, credit-a, echocardiogram, heart-h, hepatitis, hypothyroid, labor, labor-d, mushroom, primary-tumor, sick-euthyroid, soybean, vote, vote-1*

[2] The amputed attributes were *checking_status, duration, credit_history* for *credit-g*, *bxqsq, rimmx, wkna8* for *krkp*, and *intensity-mean, rawred-mean, hue-mean* for *segment*.

5 Results

In this section, we summarize our results. In section 5.1, we will start with the results on the amputed datasets, for which we can measure not only accuracy but also the degree to which increasing levels of missing values leads to a decrease in an attribute's estimated quality. Results on 24 real-world datasets will be discussed in section 5.2, before we conclude with a brief discussion of run-time differences between the methods (section 5.3).

5.1 Amputed datasets

Three datasets without missing values have been prepared as described in section 4. A key advantage of the artificial creation of the missing values in these datasets is that it facilitates having a closer look at the characteristics of the learnt models. In particular, we will consider the fraction of *amputed conditions*, i.e., rule conditions which test one of the amputed attributes. This gives an impression to which degree the classification of an example depends on these attributes.

It turns out that each of the three dataset families reveals its own specific properties. But before having a look at the results we should think about which results are to be expected. Regarding the fraction of amputed conditions, the most natural behaviour would probably be a gradual reduction of those tests according to the gradual decrease of the attributes' quality. On the other hand, the effects on the achievable accuracy should vastly depend on what Saar-Tsechansky and Provost (2007) introduced as *feature imputability* (FI). If the FI is high, i.e. the attribute's value is more or less redundant, the tests on the attribute can probably be replaced by tests on other attributes without a significant decrease in accuracy. In contrast to that, if the FI is rather low the information contained in the knocked-out attribute can hardly be replaced and thus the accuracy is likely to suffer.

5.1.1 The credit-g dataset

If we look at the results on the CREDIT-G dataset (first row of figure 5.1), we have to notice that the learner generally performs poorly on this dataset, being significantly outperformed even by the *ZeroRule*-algorithm. Against this background the results should be handled with the given care. Despite this they can still provide interesting insights. At a first glance the accuracy curves (figure 5.1.1) seem to be quite volatile. Though, three remarkable observations can be made:

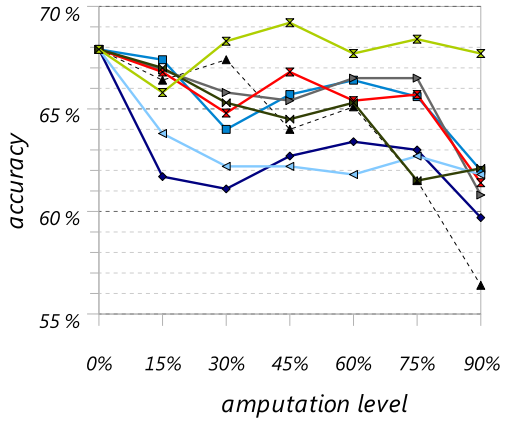
First, the *Delete* strategy performs relatively well – at a amputation level of 30% the accuracy is hardly worse than that achieved on the complete data. Only at very high amputation levels the accuracy clearly suffers. This also illustrates that even *Delete* can compete with the other strategies in case of heavy overfitting to the training data when the learner benefits from using less examples.

Secondly, both the *Pessimistic* and the *Any* strategy are clearly outperformed by all other strategies. On the other hand, *Distributed* is obviously performing best among all strategies, hardly losing any accuracy in average. The accuracy curves of both *Pessimistic/Any* and *Distributed* notably correspond to the fraction of amputed conditions in the learnt ruleset. As shown in figure 5.1.2, the influence of the amputed attributes is dramatically reduced by *Pessimistic* and *Any*, whereas the *Distributed* strategy even increases the fraction of amputed conditions.

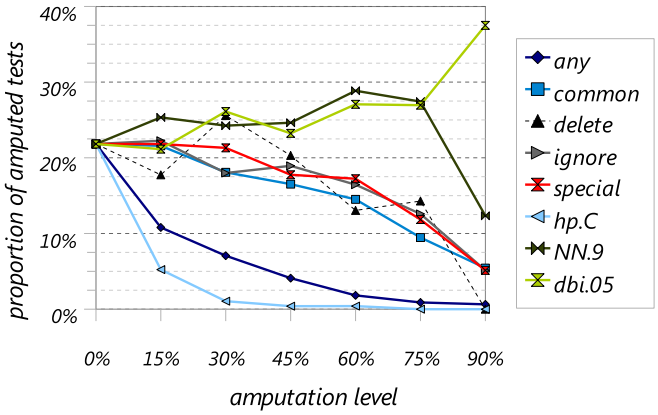
5.1.2 The KRKP dataset

The results of the learner on the *KRKP* dataset (second row of figure 5.1) are considerably better, achieving a 99% accuracy, compared to only 53% with *ZeroRule*. It turns out that the characteristics of this dataset actually are quite different, which is reflected in the course of accuracy in terms of significant losses coming along with the increasing missing-rates (figure 5.1.3). In comparison to the *Delete* strategy, the other strategies can be divided into two groups according to their accuracy: On the one hand, *Common*, *Ignore*, *Special*, *Distributed* and *Predicted* consistently outperform the *Delete* strategy, achieving very similar accuracies for almost all amputation levels. On the other hand, *Any* and *Pessimistic* are clearly outperformed by *Delete* on the lower amputation levels.

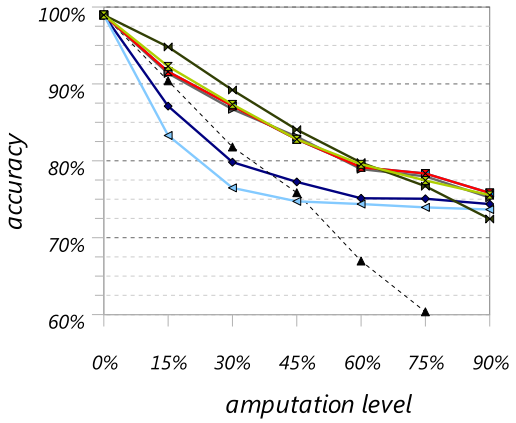
The impact of knocking out values on the learnt models is quite different from the other amputed datasets. It stands out that most strategies learn much larger models after missing values are introduced (compared to the original model, as shown in table 5.1). Also, all strategies besides *Delete* and *Pessimistic* increase the fraction of amputed tests at the beginning (figure 5.1.4). This clearly indicates that the three amputed attributes are really hard to replace in this dataset.



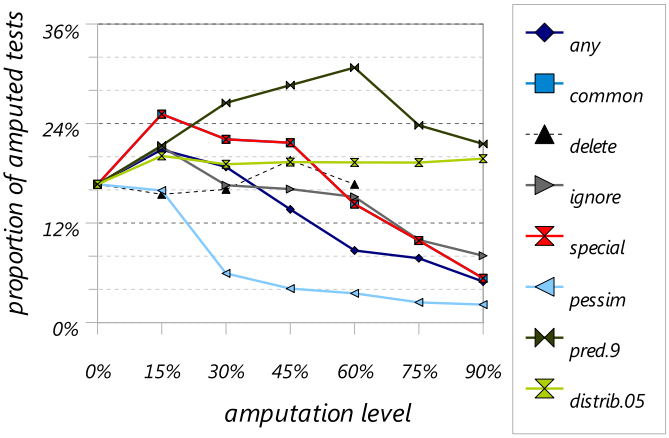
5.1.1: achieved accuracies (credit-g)



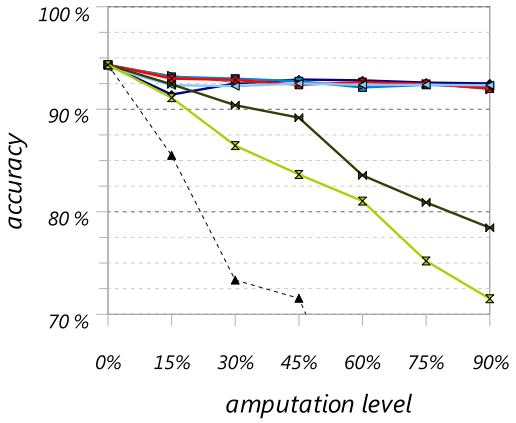
5.1.2: proportion of amputed conditions in the induced models (credit-g)



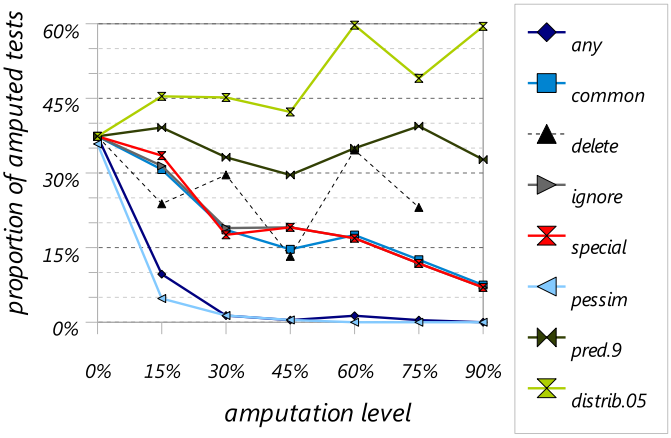
5.1.3: achieved accuracies (KRKP)



5.1.4: proportion of amputed conditions in the induced models (KRKP)



5.1.5: achieved accuracies (segment)

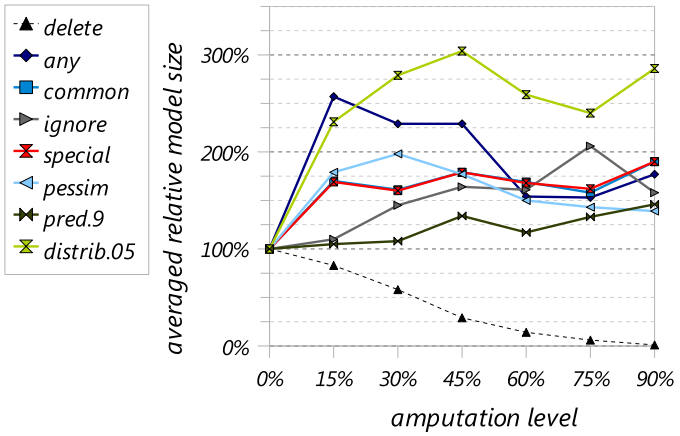


5.1.6: proportion of amputed conditions in the induced models (segment)

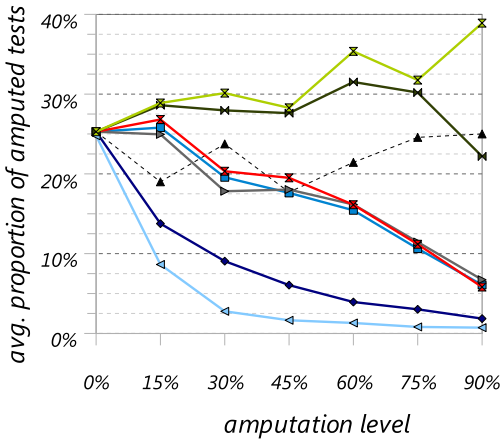
Figure 5.1: Influence of increasing missing-rates on the models learnt on the CREDIT-G (top), KRKP (middle), and SEGMENT (bottom) dataset family

m	*Delete*	*Ignore*	*Predicted*	*Common*	*Special*	*Pessimistic*	*Distributed*	*Any*
15%	-30%	+9%	+19%	+186%	+186%	+191%	+391%	+414%
30%	-41%	+106%	+34%	+149%	+149%	+255%	+549%	+344%

Table 5.1: Size of the models built on the KRKP dataset – increase/decrease (with respect to the number of conditions) compared to the model learnt on the original dataset, for two different amputation levels *m*.



5.2.1: average relative size of the models (in terms of #conditions, relative to the respective base-model)



5.2.2: average proportions of amputed conditions

Figure 5.2: Averaged properties of the models learnt on the three amputed dataset-families

On the other hand, on this dataset both the fraction of amputed tests and the model-size are no suitable measures for the strategies' quality – at least they cannot explain the poor performance of *Any*.

5.1.3 The segment dataset

Just like on the KRKP data the SeCo-learner generally performs well on the SEGMENT dataset (last row of figure 5.1). Hence, the *Delete* strategy cannot benefit from the reduction of the training set and again suffers a significant loss in accuracy. However, the accuracy-trends of the other strategies differ a lot from the previous two amputed datasets. All strategies except from *Predicted* and *Distributed* can almost keep the accuracy achieved on the original data whereas *Predicted* and *Distributed* lose up to 16% and 23% , respectively. So the FI obviously is very high in this case, thus the information contained in the missing attributes can be almost entirely substituted by using the other attributes. The fundamental difference regarding the accuracy is also reflected in the characteristics of the learnt models. The two underperforming strategies use a more or less constant (*Predicted*) or even increasing (*Distributed*) fraction of amputed conditions for classifying examples – which has certain drawbacks in cases of high FI. Moreover the size of the models learnt with these strategies gets smaller the more attribute-values are knocked-out. In contrast to that the models learnt with the other – well-performing – strategies get a bit larger with increasing amputation level.

The impact of these strategies on the fraction of amputed conditions is slightly different, though. Again, *Any* and *Pessimistic* rapidly reduce this fraction – which, in contrast to the previous two datasets, works on this dataset. On the contrary, *Common*, *Ignore* and *Special* reduce the fraction rather gradually, which should rather correspond to the "real" remaining quality of the amputed attributes.

5.1.4 Summary: amputed datasets

All in all it stands out that only the three strategies *Ignore*, *Special* and *Common* perform equally well on all three dataset-families. This observation remarkably reflects the fact that these strategies are the ones exhibiting the most natural behaviour towards missing values by gradually reducing the influence of the amputed attributes according to the fraction of removed values (as depicted in figure 5.2.2).

On the other hand, both the *Pessimistic* and the *Any* strategy seem to enforce a rapid decrease in the use of amputed conditions. At a amputation level of 30% the average fraction of amputed conditions has already dropped to merely 9% (*Any*) or even 3% (*Pessimistic*), respectively, compared to an average of 25% on the original datasets. While in the

case of *Pessimistic* the reduction is explicitly enforced by construction, this cannot be stated for *Any* where it is rather likely to be a – surprisingly strong – side-effect of the lowered selectivity of the amputed conditions (going along with the increasing coverage of example-noise). Unfortunately, this reduction is apparently hardly related to the remaining quality and importance of the attributes with respect to the models' predictive power, because in two out of three cases both strategies are outperformed by *Delete* on the lower amputation levels.

The opposite behaviour regarding the utilisation of amputed conditions could be observed with *Distributed* and *Predicted*. Both the strategies kept the fraction of amputed tests constant or even increased it on all three datasets. In particular, the fraction has not even been decreased in case the information could have actually been replaced by other attributes. This kind of bias is apparently favoured by the construction-algorithm of the amputed datasets, but the results on the segment-dataset clearly indicate the inferiority of this behaviour in cases of high FI.

So *Distributed/Predicted* and *Pessimistic/Any* reveal somehow contrary characteristics – whereas the first group sticks to the amputed attributes even if they could be replaced by known information, the latter abstain from using amputed attributes, partly putting up with a considerable overhead in theory size (cf. table 5.1), even if the respective attributes cannot be replaced equivalently. Consequently, these two pairs do not perform equally (well) on any of the three dataset-families.

But of course it has to be taken into account that three datasets can never be representative and thus the results shall primarily provide some insights in how the different strategies actually work and how they might perform under certain conditions. We also must note that the algorithm used for generating the datasets has (intentionally) been biased towards removing valuable attributes, so that we could study the effect of missing values on important attributes. It is questionable whether this is a realistic model for missing value distributions, and thus it is unclear to what extent these results can be generalized to datasets with real missing values. This will be investigated in the next section.

5.2 Real datasets

The diversity of the dataset characteristics is, of course, much higher on the real data, if only because of the larger number of datasets. Moreover, the distribution of the missing values will not suffer from an inevitable bias caused by the model behind the algorithm for generating the artificial missing values. Against this background, a one-to-one transfer of the previous results cannot be expected.

The studies on the real datasets have been performed with both the learner's default configuration using the *Laplace-estimate* as evaluation heuristic and an alternative configuration using *m-estimate* for evaluating the candidate rules. We only present aggregated results over all datasets. A detailed listing of the respective accuracies can be found in the appendix of (Wohlrab, 2009).

5.2.1 Results with the *Laplace*-heuristic

Figure 5.3 shows the average differences to the median accuracy of the eight strategies with a rule learner that uses the *Laplace*-heuristic as a search strategy. Six out of the eight strategies are very close (±0.5%) to the median-value – *Pessimistic* as the apparently best strategy achieves an average of only +0.41% whereas *Any* at the 6^{th} position still gets −0.45%. Only *Distributed* and – not surprisingly – *Delete* are clearly outperformed by the other strategies.

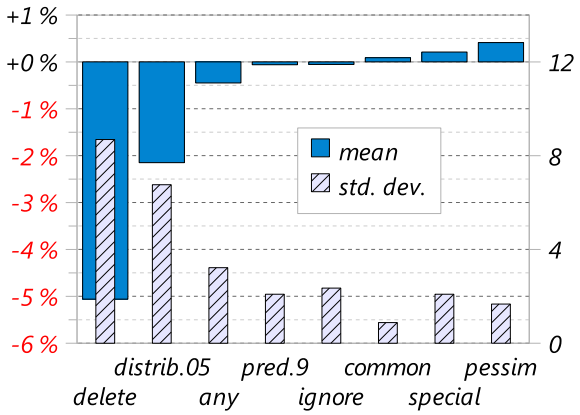
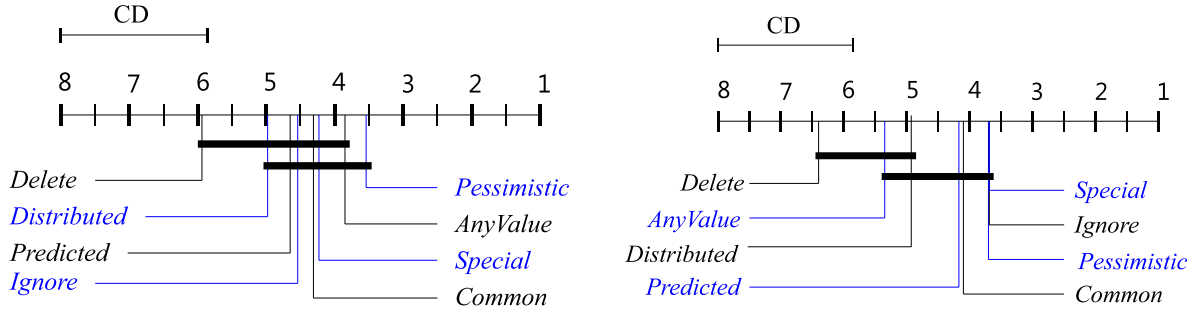


Figure 5.3: Average differences to the median accuracy for the different strategies on the real datasets

	Delete	*Distributed*	*Predicted*	*Ignore*	*Common*	*Special*	*Any*	*Pessimistic*
Laplace								
#winner	2	4	3	4	3	6	4	7
avg. rank	5.9	5.0	4.6	4.5	4.3	4.2	3.9	3.5
m-estimate								
#winner	1	2	4	7	5	6	3	4
avg. rank	6.4	5.4	4.9	4.2	4.1	3.7	3.7	3.7

Table 5.2: Number of "won" datasets and average rank of the different strategies, using the Laplace-estimate and the m-estimate, respectively, as evaluation-heuristic



5.4.1: using the Laplace heuristic

5.4.2: using the m-estimate heuristic

Figure 5.4: Nemenyi-diagrams (for $\alpha = 0.05$), depicting the average ranks and the $CD_{0.05}$ – the strategies connected by a bar do *not* differ significantly

Pairwise t-tests reveal that all strategies besides *Distributed* perform significantly better than *Delete* (at a level of α=0.01). By relaxing the level of significance to 0.075 it becomes possible to also tell *Distributed* apart from *Delete*, but no further differentiation can be made.

Due to the large number of pairwise tests the overall probability of an error is effectively larger than the α used in the respective tests. Thus it might be preferable to ensure a globally valid significance level, such as the Friedman-F-test (Demšar, 2006; Iman and Davenport, 1980), a non-parametric rank-based test which compares all strategies at the same time. In contrast to the previously performed t-test or an ANOVA, it does not make any assumptions on the distribution of the data. Moreover the results presented by Demšar (2006) indicate that in practical ML-applications the Friedman-F-test does not have less power than an ANOVA.

The Friedman-F-statistic is F-distributed with 7 and 161 degrees of freedom. The test yields a value of 2.51 (according to the average ranks listed in table 5.2) which allows us to reject the null hypothesis that the choice of the strategy has no effect at all at a level of α=0.05. After gaining this insight we can now try to identify pairwise differences by the Nemenyi post-hoc test. This tests provides a *critical difference* CD_α, which is required so that two strategies apart can be assumed to be different with a given error-probability α.

For k=8 classifiers and N=24 datasets, the critical differences for $\alpha = 0.1$ and $\alpha = 0.05$ are $CD_{0.1}$=1.97 and $CD_{0.05}$=2.14, respectively. So only *Pessimistic* (for α=0.05) and *Any* (for α=0.1) can be identified as performing significantly better than *Delete*. All other strategies perform too similar to recognise significant differences by this test. Figure 5.4.1 shows the respective CD-diagram.

What makes it so hard to detect global differences between the strategies is that even the strategies that perform well on average perform rather poorly on some datasets, whereas even the strategies performing rather poor in average also perform very well on some datasets. This is also reflected in the quite even distribution of the "winners" among the strategies, as shown in table 5.2.

Another problem is that about one third of the considered datasets have only a few missing values – which makes the particular strategies behave very similar as they, of course, can only differ in the handling of examples having missing values. Hence, to a large degree, the strategies' order on these datasets is prone to be determined by chance. As a rank-test is not sensitive to the absolute accuracy-differences, all datasets have the same influence on the overall result which gives quite a high weight to the "random" ordering on those datasets (compared to tests based on the accuracy-differences). Thus rank-based tests like the Friedman-F-test and the Nemenyi-test are likely to be less powerful in the scenario considered here.

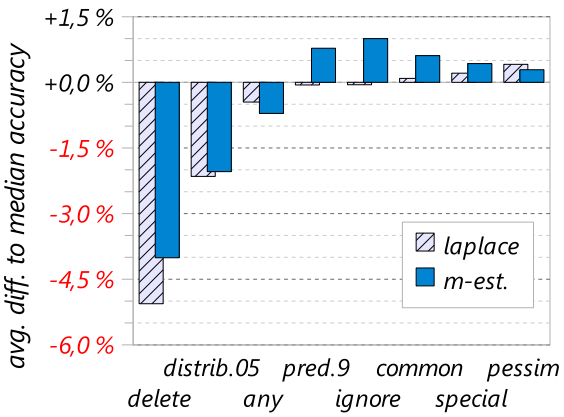


Figure 5.5: Differences to the median-accuracy achieved with Laplace- and m-estimate, respectively

5.2.2 Results with the *m-estimate*

For the sake of validation, we also ran the SeCo-learner using the m-estimate as the evaluation-heuristic. We used a value of $m = 22.466$ as recommended by Janssen and Fürnkranz (2008) for this learner. By this means, the averaged accuracy of all applied strategies increased from 0.97% (*Distributed.05*) up to 2.46% (*Ignore*). Thus, the comparison of the strategies to the median does not yield substantially different results, as can be seen from figure 5.5. The only noticable deviation is the slightly increased difference between *Any* and the top five strategies. This is also reflected in the average ranks – here *Any* loses 1.5 positions in average, dropping back from an average rank of 3.9 (position 2) to only 5.4 (position 6).

So there seems to be a more distinct separation between "good" and "not so good" strategies. This impression is also reflected by the Friedman-F-test – the respective statistic yields a value of 4.44 which allows for rejecting the null-hypothesis even at a level of α=0.0005. The CDs provided by the Nemenyi-test stay the same as before – hence the five top-ranked strategies (namely *Predicted*, *Common*, *Special*, *Ignore*, *Pessimistic*) perform significantly better than *Delete* at a level of 0.05 (figure 5.4.2).

5.2.3 Varying the Parameter of the Distributed Value Strategy

Recall that the *Distributed* value strategy has a parameter, which specifies a minimum instance weight, which specifies that fractions of instances below this threshold are ignored for classification. We evaluated seven different configurations with minimum instance weights (MIWs) between 0.01 and 0.95. With respect to two basic characteristics of the learnt models, the average size (#conditions) and rule-complexity (#conditions per rule), a very clear trend can be observed: with decreasing MIW, the models become larger and the rules more complex (table 5.3a).

The specified MIW is also almost perfectly reflected in the average ranks of the particular *Distributed*-variants, which are monotonously increasing from 2.85 for *Distributed.01* up to 5.13 for *Distributed.95* (table 5.3b). The overall differences are highly significant according to the Friedman-test (F_F=5.44, F-distributed with 6 and 132 degrees of freedom). This allows for applying post-hoc tests in order to detect pairwise differences. The Nemenyi-test yields a CD of 1.88 for α=0.05, hence at least *Distributed.01/.05* can be identified as performing significantly better than *Distributed.60/.95*. When regarding *Distributed.01* as the base-classifier the Bonferroni-Dunn-test additionally detects a significant difference to *Distributed.40* (CD=1.68).

The influence of the MIW-parameter can also be recognised with respect to the achieved accuracies. Compared to the default configuration with a MIW of 0.05, an increase of this parameter corresponds with a continuous reduction of the average accuracy. A further reduction of the MIW, however, does not lead to a higher accuracy in average – so at this point the effort put into the more detailed representation of the attributes' distribution does no longer pay off. Maybe because the strong splitting of examples facilitates an overfitting to the training data, which might be indicated by the previously mentioned effects on the complexity of the learnt models.

	Distributed.01	*Distributed.05*	*Distributed.10*	*Distributed.25*	*Distributed.40*	*Distributed.60*	*Distributed.95*
#cond	+12,83	+7,96	+6,48	-0,13	-6,43	-9,26	-18,00
#cond/rule	+0,18	+0,08	+0,05	-0,02	+0,03	-0,12	-0,32

(a) Properties of the learnt models – number of conditions in total and per rule (averaged differences to the median)

	Distributed.01	*Distributed.05*	*Distributed.10*	*Distributed.25*	*Distributed.40*	*Distributed.60*	*Distributed.95*
#winner	10	7	5	5	2	3	4
avg. rank	2.8	3.0	3.6	3.7	4.6	5.1	5.1

(b) Number of "won" datasets and average rank

Table 5.3: Evaluating the influence of the *Distributed*-strategy's MIW-parameter

	Predicted.3	*Predicted.5*	*Predicted.9*	*Predicted.15*
accuracy	+0.17	-0.33	+0.27	-0.64
rank	2.1	3.0	2.5	2.5

Table 5.4: Evaluating the influence of the neighbourhood's size – average differences to the median-accuracy and average rank of the *Predicted*-variants

5.2.4 Varying the Parameter of the Predicted Value Strategy

For the *Predicted* value strategy, we employed a nearest-neighbor classifier using `LinearNNSearch` as the search-algorithm. The parameter that could be varied effectively was the number of neighbours considered. However, the average differences between the particular variants (with respect both to the accuracies – see table 5.4 – and the models' properties) are rather small and without a clear tendency. Nevertheless, in a few cases a difference of up to 14% in accuracy could be observed. Although the Friedman-test rejects the null-hypothesis that the neighbourhood's size has no influence, it is not possible to detect a systematic trend or any significant pairwise difference.

5.3 Runtime

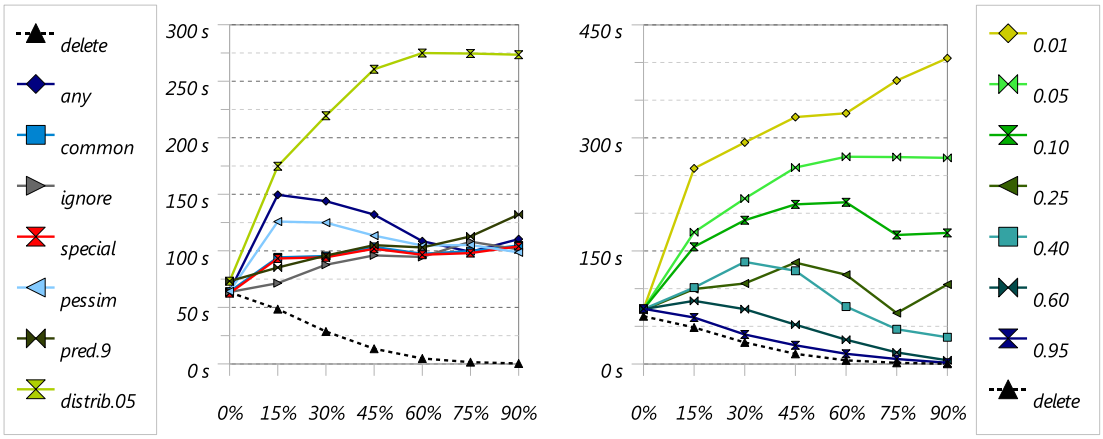
In general, the runtimes of learning-algorithms are less important, compared to the accuracies. Moreover the SeCo-learner evaluated in this paper is implemented in Java and also uses the Java-based Weka-framework and hence suffers from the JRE-typical runtime-variations. So the measured runtimes shall only give a rough impression about the order of magnitude. In this section only the averaged results on the three families of amputed datasets are taken into account – as shown in figure 5.6.1.

Not surprisingly, the by far lowest runtime is required by the *Delete* strategy. This can be easily traced back to the extreme reduction of the data – even on the first knock-out level of 15%, with three affected attributes almost 40% of the examples have at least one knocked-out value and are thus removed from the dataset. In contrary, the longest runtimes are continuously required by the *Distributed*-strategy. In particular the absence of nominal attribute values is likely to prolong the runtime as the splitting of the example effectively increases the number of training examples independent of whether the missing attribute is ever tested or not.

What stands out is the reduplication of runtimes with *Any* and *Pessimistic* on the first amputation level compared to the original dataset – this can be explained by these strategies' property to rapidly decrease the fraction of amputed conditions which has to be "paid" with a significant size-overhead. Hence the runtimes converge together with the models' properties at the higher amputation levels. Conspicuously inconspicuous, however, are the runtimes of the particular *Predicted*-variants – unlike one might expect from neighbourhood-based approaches.

If we take a closer look at the runtimes of the different *Distributed*-variants again (figure 5.6.2), we can observe the – expected – convergency of *Distributed* towards *Delete* with respect to the runtime, going along with increasing MIW. For the sake of comparability, the runtimes of *Distributed.25* are most similar to those of most other strategies depicted in figure 5.6.1.

The choice of the MIW influences the runtime of the *Distributed* strategy by trading-off two opposed effects – on one hand the training set increases due to the splitting of examples (expanding), on the other hand the number of examples decreases by discarding those examples whose weight drops below the MIW (shrinking). This trade-off can be observed very well on the run of *Distributed.40*'s runtime-curve – the expanding effect outweighs up to an amputation level of 30%, but from the 45%-level on the tables turn and the shrinking-effect prevails, leading to a continuous reduction of the runtime required.



5.6.1: standard strategies

5.6.2: Distributed-variants

Figure 5.6: Average runtimes measured on the amputed datasets

6 Related Work

Bruha and Franek (1996) performed a similar study but with a more restricted set of strategies (their strategies were essentially equivalent to the *Delete*, *Ignore*, *Common*, *Any* and *Distributed Value* strategies evaluated within this paper) and on a smaller set of datasets. In accordance to our findings, they concluded that the *Delete* routine is outperformed by all other routines and is the only routine whose use is generally discouraged by the authors. They further divide the other routines into two groups. The first – less successful – group is formed by *Distributed Value* and *Common*, whereas *Ignore* and *Any* are identified as the most successful routines. This further subdivision, however, is not supported by the results observed here. Indeed the *Distributed Value* strategy performed less successful but firstly, the difference has not been big enough in order to be significant and secondly, its performance was more similar to *Any* than to *Common*.

Moreover, Bruha and Franek observe that different strategies seem to perform well for decision-tree learners on one and separate-and-conquer rule learners on the other hand. In contrast to a separate-and-conquer learner, testing an attribute with (many) unknown values is quite problematic for a decision-tree learner because the examples are "trapped" in the respective subtree and thus have to be judged based on that attribute, whereas a separate-and-conquer learner can opt for not judging certain examples based on a particular attribute/test. This is particularly interesting as a lot of research on the field of how to handle missing attribute-values has been done with decision trees. For example, Quinlan (1989) compared numerous configurations for treating missing values for the C4.5 algorithm. He concluded that the splitting of examples (comparable to the *Distributed Value* strategy) clearly outperforms all other approaches like attempting to determine the most likely outcome of a test. It seems obvious that this strategy is better suited for decision-tree learning, where each example has to be classified in one branch, whereas in rule learning it can always be passed to the next rule (or to the final default rule).

A fundamentally different approach for handling missing attribute values – in particular at prediction-time – has been introduced by Saar-Tsechansky and Provost (2007), who suggest the use of *"reduced models"* (*RM*). The idea of the *RM*-approach is to judge every example only by such a model which has been trained using only those attributes actually specified in the respective example. In order to ensure this, a dedicated model is trained for every pattern of missing values occurring in the data (which has to be classified) using the entire available training data for each model. This approach has been compared to variants of the *Predicted Value* and *Distributed Value* strategies for the C4.5 decision tree learner, and was found to outperform both of them consistently (and sometimes even by a large margin).

In order to explain this striking performance of the *RM*s, Saar-Tsechansky and Provost (2007) introduce the concept of *feature imputability* (FI), which effectively measures an attribute's redundancy, for characterising attributes with missing values. They argue that both *Predicted Value* and *Distributed Value* could only succeed in cases of low FI (*Distributed Value*) or high FI (*Predicted Value*), respectively, whereas *RM*s should perform well in both cases and are therefore much more robust regarding the datasets' characteristics.

This finding is quite interesting as none of the strategies considered here turned out to be suitable for all (or at least the vast majority of) datasets. Against this background the utilisation of *RM*s could be a promising approach for separate-and-conquer learners and certainly deserve a closer examination. As the reduced models work on a meta-level they can be applied to any sort of inductive algorithm just out of the box.

Nevertheless, the striking performance observed with decision trees is not likely to transfer one-to-one to a separate-and-conquer learner, because the idea behind the reduced models – judging examples only by those attribute actually specified – is not so different from what the *Ignore* strategy already does. So one could argue that the particular rules effectively are nothing but "reduced models" as they apply only to those examples not lacking any of the attributes tested in the respective rule. Thus, we do not expect a large difference between *reduced models* and *Ignore*.

7 Conclusion

Not surprisingly, we have observed that each of the studied strategies has its particular strengths and weaknesses, making them perform well on some datasets and poorly on others. Against this background it is, of course, hard to detect clear-cut differences. Thus it is also hard to predict whether a particular strategy is suitable for a particular dataset or not. The only statistically backed general conclusion that can be drawn is not to apply the *Delete* strategy which significantly underperforms most of the other strategies.

If there is another general trend – then it is the merely surprising insight that the choice of the strategy for handling missing attribute values is less important on datasets with less missing values. But whereas the differences get larger on datasets with many missing values, the number of such datasets (considered for this paper) was too small to get reliable results.

A key result of our work that cannot be found in previous works is the quantification of the bias of the different strategies towards or against the use of attributes with unknown values. While the *Pessimistic* and *Any* strategies implement a bias against the use of such amputed attributes, the *Distributed* and *Predicted* value strategies, which try to replace unknown values with educated guesses, seem to maintain a potential preference for these attributes.

Interestingly, the results observed on the amputed and real datasets were inconsistent, in particular with respect to the performance of the *Pessimistic* strategy. Whereas this strategy's bias against the use of attributes with missing values turned out to be problematic on the datasets where we deliberately corrupted the three most discriminative attributes, it achieved the highest average accuracy on the real datasets. However, the *Predicted Value* strategy, which performed contrary on the amputed data, achieved about the same average accuracy on the real datasets. The obvious explanation is that the performance on the datasets that we used for our experiments does not depend as strongly on a few attributes as in the amputed datasets where we deliberately disturbed only the three most discriminative attributes. This can either mean that in most datasets a variety of different attributes can be used to form good explanations (so that a bias against some of them does not harm the performance too much), or that the missing values in these datasets are mostly concentrated on less important attributes.

The fact that there is not a single "striking" strategy suggests that a learner should have the ability to select the appropriate strategy for handling missing values dynamically based on the characteristics of the respective dataset. In this context it is certainly an interesting observation that *Pessimistic* and *Special* together "win" half of the datasets, being particularly successful on those datasets with many missing values. Whereas both the strategies perform similarly well on average, they behave quite complementary on the particular datasets, achieving an average rank of only 2.12, which is almost 1.5 ranks better than *Pessimistic* alone (cf. table 5.2). The average distance to the respectively best strategy in terms of accuracy is only 0.91% with a median of only 0.1%. So, whereas one should not expect a single strategy to perform well on most datasets, an appropriate combination of strategies with somehow complementary properties might be a promising approach for reliably achieving good results. Further research is necessary to clarify this question.

Acknowledgements

We would like to thank Nada Lavrač and Dragan Gamberger for interesting discussions on their pessimistic value strategy. This research was supported by the *German Science Foundation (DFG)* under grant FU 580/2.

Bibliography

Ivan Bruha and Frantisek Franek. Comparison of various routines for unknown attribute value processing: The covering paradigm. *International Journal of Pattern Recognition and Artificial Intelligence*, 10(8):939–955, 1996. 3, 17

Peter Clark and Robin Boswell. Rule induction with CN2: Some recent improvements. In *Proceedings of the 5th European Working Session on Learning (EWSL-91)*, pages 151–163, Porto, Portugal, 1991. Springer-Verlag. 6

Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989. 3, 6, 8

William W. Cohen. Fast effective rule induction. In A. Prieditis and S. Russell, editors, *Proceedings of the 12th International Conference on Machine Learning (ML-95)*, pages 115–123, Lake Tahoe, CA, 1995. Morgan Kaufmann. 3, 5

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7: 1–30, 2006. 13

Dragan Gamberger, Nada Lavrač, and Johannes Fürnkranz. Handling unknown and imprecise attribute values in propositional rule learning: A feature-based approach. In Tu-Bao Ho and Zhi-Hua Zhou, editors, *Proceedings of the 10th Pacific Rim International Conference on Artificial Intelligence (PRICAI-08)*, pages 636–645, Hanoi, Vietnam, 2008. Springer-Verlag. 6

Seth Hettich, Catherine L. Blake, and Christopher J. Merz. UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 1998. Department of Information and Computer Science, University of California at Irvine, Irvine CA. 8

Ronald L. Iman and James M. Davenport. Approximations in the critical region of the Friedman statistic. *Communications in Statistics — Theory and Methods*, 9(6):571–595, 1980. 13

Frederik Janssen and Johannes Fürnkranz. An empirical investigation of the trade-off between consistency and coverage in rule learning heuristics. In J.-F. Boulicaut, M. Berthold, and T. Horváth, editors, *Proceedings of the 11th International Conference on Discovery Science (DS-08)*, pages 40–51, Budapest, Hungary, 2008. Springer-Verlag. 8, 14

Nada Lavrač, Johannes Fürnkranz, and Dragan Gamberger. Explicit feature construction and manipulation for covering rule learning algorithms. In Jacek Koronacki, Slawomir T. Wirzchon, Zbigniew Ras, and Janusz Kacprzyk, editors, *Recent Advances in Machine Learning — Dedicated to the Memory of Ryszard Michalski*. Springer-Verlag, To appear. 6

J. Ross Quinlan. Unknown attribute values in induction. In *Proceedings of the 6th International Workshop on Machine Learning (ML-89)*, pages 164–168, 1989. 6, 17

Maytal Saar-Tsechansky and Foster Provost. Handling missing values when applying classification models. *Journal of Machine Learning Research*, 8:1625–1657, 2007. 9, 17

Ian H. Witten and Eibe Frank. *Data Mining — Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers, 2nd edition, 2005. URL `http://www.cs.waikato.ac.nz/~ml/weka/`. 8

Lars Wohlrab. Comparison of different methods for handling missing attribute values in the SeCo rule learner. Independent Study Project, Knowledge Engineering Group, TU Darmstadt, 2009. In German. 8, 12