Comparison of various routines
for unknown attribute value processing: the covering paradigm

Ivan Bruha and Frantisek Franek

McMaster University, Dept. Computer Science and Systems
Hamilton, Ont., Canada L8S4K1
Email: {bruha | franya}@mcmaster.ca

Abstract

Simple inductive learning algorithms assume that all attribute values are available. The well-known Quinlan's paper [Qui89] discusses quite a few routines for processing of unknown attribute values in the TDIDT family and analyzes seven of them.

This paper introduces five routines for processing of unknown attribute values that have been designed for the CN4 learning algorithm, a large extension of the well-known CN2. Both algorithms CN2 and CN4 induce lists of decision rules from examples applying the covering paradigm.

CN2 offers two ways for processing of unknown attribute values. The CN4's five routines differ in style of matching complexes with examples (objects) that involve unknown attribute values. The definition of matching is discussed in detail in the paper. Strategy of unknown value processing is described both for learning and classification phases in individual routines.

The results of experiments with various percentages of unknown attribute values on real-world (mostly medical) data are presented and performances of all five routines are compared.

Keywords: machine learning, rule induction, unknown attribute value processing, matching

1.   Introduction

When inducing decision trees or decision rules from real-world data, many various aspects are to be taken into account. One important aspect in particular is the processing of *unknown* attribute values. This topic has been discussed and analyzed by several researchers in the field of machine learning [Qui86], [CKB87], [BB92].

There are a few directions in which unknown attribute values as well as the corresponding routines for their processing may be studied and designed. First, the *source of 'unknownness'* should be investigated; there are several such sources [Ko92], [C92]:
● a value is *missing* because it was forgotten or lost,
● a certain attribute is *not applicable* for a given object , e.g., it does not exist for a given observation,
● an attribute value is *irrelevant* in a given context,

1

● for a given observation, a designer of a training database does not care about a value of a certain attribute (so-called *dont-care* value).

The second direction is studying the effects of different *places* in the algorithm where the routines for unknown value processing are situated. For instance, [Qui89] conceders three different places: attribute test evaluation, partitioning of training set, and classification, and all their mutual combinations. The third, formulas for *matching objects* (examples) containing unknown attribute values *with selectors* (and complexes) of decision trees and decision rules are important to be defined, since these routines differ particularly in this fashion.

[Qui89] surveys and investigates quite a few techniques for unknown attribute values processing for the TDIDT (Top Down Induction Decision Trees) family and analyzes seven combinations of their placement in the learning algorithm.

The aim of this paper is to portray the performance of various unknown value processing strategies in an algorithm employing the *covering* paradigm, which is rather different from the divide-and-conquer paradigm utilized by the TDIDT algorithms. These two paradigms are more or less complementary; there are problems in machine learning more suitable to one paradigm than the other and vice versa. The contribution of our paper is to demonstrate how the unknown attribute value processing may be incorporated into an algorithm based on the covering paradigm.

Following the CN2 algorithm [CN89], [CB91] in spirit, we have designed and implemented a completely new version of this inductive algorithm and incorporated various enhancements, among them five routines for unknown attribute value processing. We call this version CN4 [BK94], [KB93]. A user can choose one of the above routines, however, the current version of our inductive algorithm does not distinguish the sources of unknownness, similarly as [Qui89]. Currently we are working on extensions of this project which would address this problem.

It should be also noticed that CN2 distinguishes two sources of unknown values [Bos90], namely *missing* and *dont-care* values. The user thus has to differentiate them in a training database (by introducing '?' and '*' , respectively), but otherwise he/she cannot select matching formulas.

Section 2 of this paper surveys the methodology of the CN*x* family of covering algorithms. The ways CN4 processes unknown attribute values are presented in Section 3. Experiments exhibiting the performance of this version are discussed in Section 4, and their results are analyzed in Section 5.

2.   The CN*x* Family

The inductive algorithms of the CN*x* family comprise two main procedures: the top procedure that repeatedly controls the search for decision rules, and the search procedure that performs the beam search to find the best complex for a portion of the training set.

The inductive algorithm generates decision rules from a set of K training examples, each accompanied by its desired class C_r, r = 1,...,R . Examples 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[1]. The algorithm yields either an *ordered* or an *unordered* list of decision rules of the form

    Rule: if Cmplx then class is C_r

A complex *Cmplx* is evaluated according to a user-specified heuristic evaluation function which is one of the three: either (negative) entropy

$$NegEntr(Cmplx) = \sum_r \frac{K_r(Cmplx)}{K(Cmplx)} \log_2 \frac{K_r(Cmplx)}{K(Cmplx)} \tag{1}$$

or Laplacian criterion for expected accuracy (for the class C_r)

$$Lapl(C_r, Cmplx) = \frac{K_r(Cmplx) + 1}{K(Cmplx) + R} \tag{2}$$

or m-probability [Ces90]

$$m\text{-}prob(C_r, Cmplx) = \frac{K_r(Cmplx) + m f_r}{K(Cmplx) + m} \tag{3}$$

where $K_r(Cmplx)$ is the number of training examples of the class C_r covered by the complex *Cmplx*, called *class-sensitive* coverage,
$K(Cmplx)$ is the total number of examples covered by the complex (*over-all* coverage),
R is the number of classes involved in the given task, and
f_r is the relative frequency of the class C_r in the entire training set.

The larger the value of the evaluation function, the better the complex. It should also be noticed that the evaluation through the entropy may be used only in the ordered mode.

The appendix comprises the flow charts of both procedures of CN*x* for a reader to find easily where the routines for unknown attribute values are positioned.

3.   Unknown Attribute Value Processing

3.1.   Survey

To deal with real-world situations, it is necessary to process incomplete data, i.e. data with unknown attribute values. Five routines for unknown-attribute-values processing have been designed for CN4. They differ in the style of their solution of the matching formula.

The following natural ways of dealing with unknown attribute values were incorporated:
(i)   ignore the example (object) with unknown values (routine *Ignore*),
(ii)   consider the unknown value as an additional regular value for a given attribute (routine *Unknown*), or
(iii)   substitute the unknown value for matching purposes by a suitable value which is either

---

[1] There exist several procedures for discretizing numerical attributes. The one implemented in CN4 is described in [BK94]. An off-line discretization has been tested, too [BB95].

3

- the most common value (routine *Common*),
- a proportional fraction (routine *Fraction*), or
- any value (routine *Anyvalue*)

of the known values of the attribute that occur in the training set.

There are three places where these routines are situated (compare Appendix):

(a) <u>learning at the search procedure</u>
- when evaluating a complex (either to compare it with the current best complex, or if the star is to be trimmed - see procedure SEARCH, steps 3 and 4 within **while** loop):

(b) <u>learning at the top procedure</u>
- when removing the covered examples from the training set (see procedure CN*x*, the first two lines of the step 2, the **until** loop);

(c) <u>classification</u>
- when matching a rule to an unseen object.

The five routines discussed below correspond roughly to the following combinations in [Qui89]: IIC (our routine *Ignore*), RUU (*Unknown*), CCC (*Common*), RFF (*Fraction*), and RAF (*Anyvalue*).

Dealing with unknown attribute values in all above places is in fact determined by matching a complex with an object. A matching procedure of a complex with a fully specified object returns the uniform solution: the object either matches the complex or not. Dilemmas arise when a partially defined object is to be matched with a complex.

3.2.   Matching

CN4 like most algorithms that process unknown values needs to know in advance (for r=1,...,R, n=1,...,N, 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 an unknown value by a suitable one, we take the desired class of the example into consideration in order not to increase the noise in the data set. On the other hand, the over-all frequencies are applied within classification.

Let us now consider an example (object) $\mathbf{x} = [x_1, ..., x_N]$ involving N attribute values and a complex
$$Cmplx = S_{q_1} \ \& \ ... \ \& \ S_{q_M}$$

where S_{q_m}, m = 1,...,M, is the m-th selector testing the j-th value V_j of the q_m-th attribute, i.e. exhibiting the form

$$x_{q_m} = V_j$$

Now we are going to define the matching of the example **x** and the complex *Cmplx*.

First, selector's *matching ratio* satisfies the following:

$$\mu(\mathbf{x}, S_{q_m}) \quad \begin{cases} = 0 & \text{if } x_{q_m} \neq V_j \\ = 1 & \text{if } x_{q_m} = V_j \\ \in [0; 1] & \text{if } x_{q_m} \text{ is unknown} \end{cases} \tag{4}$$

A particular value of the matching ratio is determined by the selected routine for unknown value processing. The first two cases correspond to the 'standard' way of matching.

The *matching ratio* of the complex *Cmplx* may be determined in various ways. It is either equal to
- the product of matching ratios of its selectors:

$$\mu(\mathbf{x}, Cmplx) = w(\mathbf{x}) \prod_{m=1}^{M} \mu(\mathbf{x}, S_{q_m}) \tag{5}$$

- or their average:

$$\mu(\mathbf{x}, Cmplx) = \frac{w(\mathbf{x})}{M} \sum_{m=1}^{M} \mu(\mathbf{x}, S_{q_m}) \tag{6}$$

where $w(\mathbf{x})$ is the weight of the example $\mathbf{x}$ (1 by default).

Second, the class-sensitive coverage of the complex *Cmplx* is expressed as the weighted sum of matching ratios of the given complex and all training examples of the class C_r

$$K_r(Cmplx) = \sum_{\mathbf{x} \in C_r} \mu(\mathbf{x}, Cmplx) \tag{7}$$

and the over-all coverage of the complex is represented by

$$K(Cmplx) = \sum_r K_r(Cmplx) \tag{8}$$

The above formulas are utilized by the heuristics (1) to (3).

3.3. The Five Routines

(A) *Routine Ignore: Ignore Unknown Values*

This strategy simply ignores examples with at least one unknown attribute value before learning. Hence, no dilemma arises when determining matching ratios within learning. However, this approach does not contribute to any enhancement of processing noisy or partly specified data.

As for classification, the unknown value does not match any regular (known) value in a rule's complex. Thus, selector's matching ratio is equal to 0 for any unknown value. Consequently, only the rules that test the regular values of an unseen object may succeed. If there is no such rule, then the default rule is applied; i.e., the object is classified as belonging to the majority class.

(B) *Routine Unknown: Unknown Value as a Regular One*

An unknown value is considered as an additional attribute value. Hence, the number of values is increased by one for each attribute that depicts an unknown value in the training set. The matching ratio of a selector comprising the test of the attribute A_n and an object with A_n unknown is equal to 1 if and only if this test is of the form $x_n = ?$ where '?' represents the unknown value.

Note that selectors corresponding to a numerical (continuous) attribute are formed by tests $x_n \in V_j$ (where V_j is a numerical interval) or $x_n = ?$.

(C)   *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. An unknown 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 $\dfrac{F_{r,n,j} + 1}{F_{n,j} + R}$ over j for the given r and n. If the maximum is reached for more than one value of A_n, then the value V_j with the greatest frequency $F_{r,n,j}$ is selected as the common value. An unknown value within classification is replaced by the *over-all common* value which maximizes $F_{n,j}$ over the subscript j . Consequently, the matching ratio yields 0 or 1, since every unknown value is substituted by a concrete known value.

The Laplacian formula utilized within learning prefers those attribute values that are more predictive for a given class in the contrary to the conventional 'maximum frequency' scheme. For instance, let an attribute have two values V_1 and V_2 , with (absolute) frequencies [4, 2] and [3, 0], respectively, above the classes C_1 and C_2 . The maximum frequency then chooses the value V_1 as the most common value, whereas the Laplacian formula prefers the value V_2 as the more predictive for the class C_1.

There are no theoretical reasons why to prefer one most common value over another if they have the same frequency. Our routine simply selects the first attribute value with the highest frequency. The order of attribute values is determined by the user, and hence, he/she can affect the value chosen if so desired. For the future extension, a subroutine that selects the most common value according to some criteria might easily be implemented.

It is virtually standard to discretize numerical values continuously distributed over a range by subdividing the range into intervals and considering all values falling into an interval to have the same discrete (categorical) value [BB95]. Such an approach, of course, is rather sensitive to the way the range is divided into intervals (categories).

In our routine, for any numerical (continuous) attribute the entire numerical range is partitioned into a pre-specified number of equal-length intervals and their frequencies are determined. An unknown value of the numerical attribute is then substituted by the mean value of the interval with the maximum frequency. Thus, these intervals (categories) play the role of discrete attribute values. Using the mean value moderates to some degree the disturbing effect of the selection of the partitioning of the range as described above.

(D)   *Routine Fraction: Split into Proportional Fractions*

(i)   Learning phase

The learning phase requires that the relative frequencies $f_{r,n,j}$ above the entire training set be known. Each example **x** of class C_r with an unknown value of a discrete attribute A_n is substituted by a collection of examples before the actual learning phase as follows: unknown 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

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

where $w(\mathbf{x})$ is the weight of the original example **x** . The weight is assigned by the designer of the training set and represents his/her subjective judgement of the importance of that particular example within the entire training set.[2] The matching ratio of the split examples is accomplished by (4) and (5) in a standard way.

If a training example involves more unknown attribute values, then the above splitting is done for each unknown value. Thus, the matching ratio may rapidly decrease. Therefore, the routine involves the star

_____

[2] It is not in the scope of this paper to discuss the methods and methodologies how to assign weights to examples in training sets, see [Qui92] for details.

methodology to avoid explosion of examples so that only a predefined number of split examples with the largest weights is used for replacement of the original example.

In case of a numerical (continuous) attribute, an example with an unknown numerical value is not divided before the learning phase. The actual splitting takes place when an example (of the class C_r) involving an unknown numerical value is tested in a newly generated rule (step 2 in the procedure CNx). The matching ratio μ of a numerical selector is determined by the relative frequency of numerical values (occurring in the class C_r) in the interval of the tested selector. The example is then split into two fragments: the portion μ which is covered and the portion 1-μ which remains uncovered.

(ii)   Classification phase

The routine *Fraction* works in two different modes within classification: ordered and unordered. The matching ratio for a selector $x_n = V_j$ is defined by (4) as $\mu = f_{n,j}$ for an unknown value of A_n. The complex's ratio for the ordered mode is specified by (5) and that for the unordered mode by (6).

To classify an unseen object in the ordered mode, the system tests the rules in a determined order and calculates the matching ratio for each rule's complex. If the unseen object is regular (without any unknown value), then the classification is carried out in the traditional way. However, if an object involves one or more unknown values, then the system tries in turn all the rules (except the last, default one):

- If a rule that matches completely the given object is reached - i.e., the matching ratio of its complex is equal to 1 - then the classification ends and the object is categorized as belonging to the class attached to this rule.
- If no rule matches completely, then the system sums up the rules' matching ratios for each class separately, and classifies the given object as belonging to the class with the maximum sum.

Formula (5) utilized in the ordered-mode classification is thus in principle similar to the token scenario developed for the decision trees derived by ID3 [Qui86], [Qui89].

The decision scheme of the unordered-mode classification looks as follows:

- If a rule that matches completely the given object is reached - i.e., its matching ratio is equal to 1 - then the classification ends and the object is categorized as belonging to the class attached to this rule.
- If no rule matches completely, then the rule whose complex yields maximum matching ratio is detected and its class is attached to the object.

Formula (6) applied in the unordered-mode classification is more or less equivalent to the *flexible matching* used e.g. in [Tor93].

Let us consider the following two rules illustrating the influence of the different formulas of matching ratios in the ordered and unordered mode:

```
Rule1:  if S₁ & S₂ then class is C₁
Rule2:  if S₁ & S₂ & S₃ then class is C₂
```

Let the selectors S_1 and S_3 match an example **x** with the ratio 1, and S_2 with 0.5 . The matching ratio (5) for both rules is 0.5, but the unordered mode prefers the rule *Rule2* since it exhibits more completely matching selectors following (6):

$$\mu(\mathbf{x}, Rule1) \;=\; \tfrac{1}{2}\,(\mu(\mathbf{x}, S_1) + \mu(\mathbf{x}, S_2)) \;=\; \tfrac{1}{2}(1 + 0.5) \;=\; 0.75$$

$$\mu(\mathbf{x}, Rule2) \;=\; \tfrac{1}{3}\,(\mu(\mathbf{x}, S_1) + \mu(\mathbf{x}, S_2) + \mu(\mathbf{x}, S_3)) \;=\; \tfrac{1}{3}(1 + 0.5 + 1) \;=\; 0.83$$

(E)   *Routine Anyvalue: Any Value Matches*

An unknown value matches any existing attribute value, both in learning and classification. Therefore, matching ratio μ of any selector is equal to 1 for any unknown value.

It should be noticed that there is no uniform scheme in machine learning for processing the 'any-value'.

- In some systems, especially TDIDT, an example with an unknown value for attribute A_n (which is to be treated as 'any-value') is replaced by J(n) examples in which the unknown value is in turn substituted by each regular value V_j, j=1,...,J(n) ; weights of new examples are equal to the weight of the original one.
- In other systems, especially those utilizing the covering paradigm, the unknown 'any-value' is substituted by any first attribute value involved in a newly generated rule when covered examples are being removed from the training set (see the second line of step 2 in the procedure CN*x*).

4.   Experiments

This section describes the experiments that have been carried out in order to compare the above five routines for unknown attribute value processing. The CN4 algorithm has been employed as a knowledge acquisition vehicle.

Each routine was run for these modes and heuristics: (i) the ordered mode with entropy, (ii) the ordered mode with Laplacian evaluation function, and (iii) the unordered mode with Laplacian function.

All combinations have been tested on four well-known AI databases. Each database has been randomly separated to two subsets (70% learning, 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 10 runs. The parameters of CN4 were set up to their default values (i.e. star size equal to 5, parameter for χ^2-distribution equal to 0.025).

The four AI databases are as follows:

- *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 [Pe93] were used for testing in the Czech Academy of Sciences, Prague, Czechland, and also in the project CN4 [BK93]. 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 to the authors of this paper 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%).

To achieve extensive and comprehensive comparison of the above routines' behaviour we decided to find how classification accuracy depends on various percentage of unknown attribute values in databases.

However, only the database *ThyroidGland* exhibits a reasonable size of 'unknownness'. Therefore, we performed three experiments with this database for various number of attributes:
- set with all 19 attributes (there is 30% unknown values),
- set with 10 most informative attributes[3] (there is 20% unknown values),
- set with 5 most informative attributes (there is 15% unknown values).

As for the remaining databases, to emulate various number of unknown values, we have run the original data through a filter which randomly changes attribute values to unknown ones. The filter procedure ('unknownizer') has the percentage of unknown attribute values as its parameter. As one can see from the following results, we have run the filter to get datasets with 5%, 10%, 15%, 20%, and 30% unknown values.

Tab. 1 comprises the average classification accuracy (in %) for the dataset *ThyroidGland* for 15%, 20%, and 30% of unknown values, and (i) ordered mode, entropy, (ii) ordered mode, Laplacian evaluation, and (iii) unordered mode, Laplacian evaluation.

Fig. 1 to 3 (and corresponding Tab. 2 to 4) depict the results for the three remaining datasets. There are five lines in each graph for each of the routine; they indicate how classification accuracy depends on the percentage of unknown attribute values. Each graph point (and each number in tables) represents an average value done from the three above modes (i), (ii), and (iii); i.e., it reflects an average of 30 separate runs.

5.   Analysis

The aim of this project was to design and experimentally compare various unknown-attribute-value processing routines for inductive algorithms utilizing the covering paradigm. The only, but widely used criterion in our experiments was the classification accuracy acquired from testing sets. However, in any experimental project like this one, the classification accuracy depends on various factors; here particularly on:
(a)   the routine for processing unknown attribute values (what was the main aim of this paper);
(b)   the "quality" of training dataset, i.e. its fuzziness, representativeness of examples, presence of contradictory examples, and the like;
(c)   parameters of the inductive algorithm CN4, namely star size, significance threshold, and the mode and heuristic functions for complex evaluation.

The issue (c) was not part of this project, since it was deeply investigated in [BK93], [BK94], [KB93]. Just to summarize the earlier results, we have not found any profound dependence of the classification accuracy and the number of decision rules induced on the star size. A higher significance threshold generally induces smaller number of decision rules which exhibit a weaker classification accuracy. The ordered mode is slightly better in its performance than the unordered one. From a theoretical viewpoint, if training data are sufficiently representative, then the algorithm with a smaller significance threshold might induce rules with

---

[3] Informativity of attributes were measured by the information gain, see e.g. [Qui86].

100% accuracy, however, the number of rules would be quite large and the rules derived would exhibit meaningless predictability.

Each dataset exhibits different magnitude of noise, distortion, and fuzziness. Since all the above routines were applied to all datasets, the noise of each set is thus more or less eliminated.

Tab. 5 exhibits the over-all results, averaged for all datasets, all sizes of 'unknownness', and all the modes. It shows not only the classification accuracy (performed on testing subsets) but also average numbers of decision rules induced (which may interpreted a simple measures for sizes of concept descriptions).

When comparing the results gained by CN4 we came to the following:

- The five routines discussed above may be separated into the three groups. The routine *Ignore* is evidently the worst strategy; we suggest not to apply it at all. The second group is formed by the routines *Common* and *Fraction* with accuracy about 1 to 2% below the third group. On average, the most successful are the strategies *Unknown* and *Anyvalue*.

- This contradicts to some extent [Qui89] which indicates that the routine *Fraction* is one of the best for processing unknown values in ID3. The explanation of this fact is based on different ways of processing examples in the covering paradigm as opposed to TDIDT: in TDIDT, all examples from the training set are eventually incorporated into the decision tree generated by the learning algorithm. On the other hand, the covering paradigm algorithm generates rules which may not cover all of the examples from the training set (since some of the examples are found not to be representable).

- The above table also indicates that the classification accuracy more or less depends on the number of rules induced. Although the parameters of CN4 (star size, significance threshold etc.) were set up identical for all the five routines, they induced quite different sizes of the concept descriptions.

- Although the routine *Unknown* is one of the 'winners' in our experiments, however, it is not clear to us how to interpret on philosophical as well as semantic level a decision rule that involves a selector with an attribute equal to '?' (unknown value).

- We have also found out that each dataset has more or less its own 'favourite' routine for processing unknown attribute values. It evidently depends on the magnitude of noise and source of unknownness in each dataset, measured e.g. by methodology in [KB91]. A promising idea seems to be to explore the most suitable routine(s) for training datasets. We are going to investigate this issue in the future.

- Another future project is to distinguish the sources of unknownness. The first attempt was done in [B95].

The covering algorithm CN4 has been written in C and runs under both Unix and MS-DOS. Anybody requesting a copy of the source code should contact the first author, preferably by Email.

Appendix

The flow charts of the learning and search procedures for the ordered mode look as follows (here T is a training set):

procedure CNx(T)
    Let *ListOfRules* be an empty list
    Until T is empty **do**
        1. Let *Cmplx* be the best complex found by the search procedure SEARCH(T) for the given set T and user-selected heuristic

2. If *Cmplx* is not nil then

Let $T' \subseteq T$ be examples covered by *Cmplx*

Let *T* become $T \setminus T'$

Add the rule `If Cmplx then class is C` to the end of *ListOfRules* where *C* is the majority class in T'

else break (quit the loop)

enddo

Add the default rule `If true then class is majority class` to the end of *ListOfRules*

Return *ListOfRules*

procedure SEARCH(*T*)

Let *Star* be a set containing just the *true* complex (that matches anything)

Let the complex *Best* be nil

Let *Sel* be the set of all possible selectors occurring in *T*

While stopping condition is not satisfied **do**

1. Let *NewStar* be an empty set
2. Specialization: Specialize each complex *p* in *Star* by adding any possible selector *s* from *Sel* (except contradictory or unchanged selectors), then evaluate the new complex *p* & *s* by user-selected heuristic and include it in the set *NewStar*
3. Searching for the best complex:

    For each complex *p* in *NewStar* **do**

    if *p* is statistically significant (i.e., its significance is greater than a predefined threshold) and is better than *Best* measured by the used-selected heuristic, then replace the current value of *Best* by *p*

    enddo
4. Trimming: If the size of *NewStar* is greater than a user-defined maximum (so-called *star size*), then remove the worst complexes from *NewStar*, again by applying the heuristic
5. Let *Star* become *NewStar*

enddo

Return *Best*

To generate an unordered list of decision rules the above principle is sequentially activated for each class separately.

References

[B95]   I. Bruha: *Unknown attribute values processing utilizing expert knowledge on attribute hierarchy*. 8th European Conf. on Machine Learning, Workshop Statistics, Machine Learning, and Knowledge Discovery in Databases, Heraklion, Crete (1995), 130-135

[BB92]   P.B. Brazdil and I. Bruha: *A note on processing missing attribute values: a modified technique*. Workshop on Machine learning, Canadian Conf. AI, Vancouver (1992)

[BB95]   P. Berka and I. Bruha: *Various discretizing procedures of numerical attributes: Empirical comparisons*. 8th European Conf. on Machine Learning, Workshop Statistics, Machine Learning, and Knowledge Discovery in Databases, Heraklion, Crete (1995), 136-141

[BK93]   I. Bruha and S. Kockova: *Quality of decision rules: empirical and statistical approaches*. Informatica, **17** (1993), 233-243

[BK94]   I. Bruha and S. Kockova: *A support for decision making: cost-sensitive learning system.* Artificial Intelligence in Medicine, 6 (1994), 67-82

[Bos90]   R. Boswell: *Manual for CN2, version 4.1.* Turing Institute, Techn. Rept. P-2145/Rab/4/1.3 (1990)

[C92]   P. Clark: private communication (1992)

[CB91]   P. Clark, R. Boswell: *Rule induction with CN2: some recent improvements.* EWSL'91, Porto (1991), 151-163

[Ces90]   B. Cestnik: *Estimating probabilities: a crucial task in machine learning.* ECAI-90 (1990)

[CKB87]   B. Cestnik, I. Kononenko, I. Bratko: *ASSISTANT 86: a knowledge-elicitation tool for sophist-icated users.* In: I. Bratko, N. Lavrac (eds.): *Progress in machine learning.* Proc. EWSL'87, Sigma Press (1987)

[CN89]   P. Clark and T. Niblett: *The CN2 induction algorithm.* Machine Learning, 3 (1989), 261-283

[KB91]   I. Kononenko and I. Bratko: *Information-based evaluation criterion for classifier's perfor-mance.* Machine Learning, 6 (1991), 67-80

[KB93]   S. Kockova and I. Bruha: *CN4: an extension of covering algorithm CN2.* AS CR, Techn. Report (1993)

[Ko92]   I. Kononenko: *Combining decisions of multiple rules.* In: B. du Boulay and V. Sgurev (eds.): Artificial Intelligence V: Methodology, Systems, Applications. Elsevier Science Publ. (1992), 87-96

[Pe93]   L. Pecen: private communication. Prague (1993)

[Qui86]   J.R. Quinlan: *Induction of decision trees*, Machine Learning, 1 (1986), 81-106

[Qui89]   J.R. Quinlan: *Unknown attribute values in ID3.* Interntl. Conf. ML (1989), 164-8

[Qui92]   J.R. Quinlan: *C4.5 programs for machine learning.* Morgan Kaufmann (1992)

[Tor93]   L. Torgo: *Controlled redundancy in incremental rule learning.* ECML-93 (1993), 185-195

	Ignore			Unknown			Common			Fraction			Anyvalue		
	(i)	(ii)	(iii)	(i)	(ii)	(iii)	(i)	(ii)	(iii)	(i)	(ii)	(iii)	(i)	(ii)	(iii)
15%	80	81	69	89	90	88	83	85	85	87	80	83	89	88	80
20%	77	77	65	90	91	84	84	85	85	86	87	83	88	92	82
30%	79	73	55	90	91	85	83	85	84	90	90	83	91	92	83

Tab. 1. Average classification accuracy (in %) for the dataset *ThyroidGland*.

	Ignore	Unknown	Common	Fraction	Anyvalue
5%	81	86	85	83	84
10%	80	85	85	81	84
15%	80	84	85	82	83
20%	80	83	83	82	83
30%	79	82	83	81	83

Tab. 2. Average classification accuracy (in %) for the dataset *BreastTumor*.

	Ignore	Unknown	Common	Fraction	Anyvalue
5%	69	75	74	72	75
10%	69	74	74	71	75
15%	67	72	71	70	73
20%	66	71	71	70	72
30%	64	71	70	69	72

Tab. 3. Average classification accuracy (in %) for the dataset *Onco*.

	Ignore	Unknown	Common	Fraction	Anyvalue
5%	69	75	74	72	75
10%	69	74	74	71	75
15%	67	72	71	70	73
20%	66	71	71	70	72
30%	64	71	70	69	72

Tab. 4. Average classification accuracy (in %) for the dataset *Soybean*.

	Ignore	*Unknown*	*Common*	*Fraction*	*Anyvalue*
classification accuracy	73.0	80.6	79.5	78.7	80.6
# rules induced	6.9	13.2	9.5	9.9	15.2

Tab. 5. Over-all classification accuracy and number of rules induced, averaged for all datasets, all sizes of 'unknownness',   and all the modes.
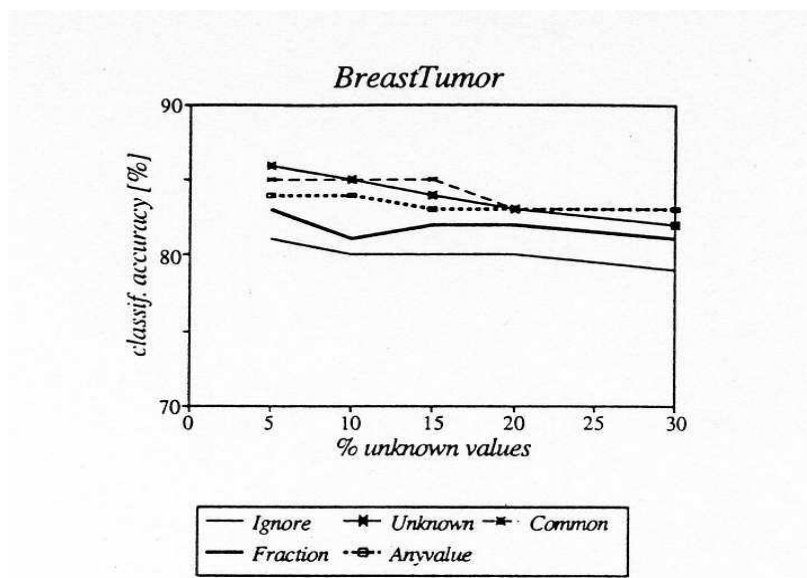

Fig. 1. Classification accuracy vs. percentage of unknown values for the five routines, performed on the dataset *BreastTumor*.. Each point represents average accuracy of the three learning modes.
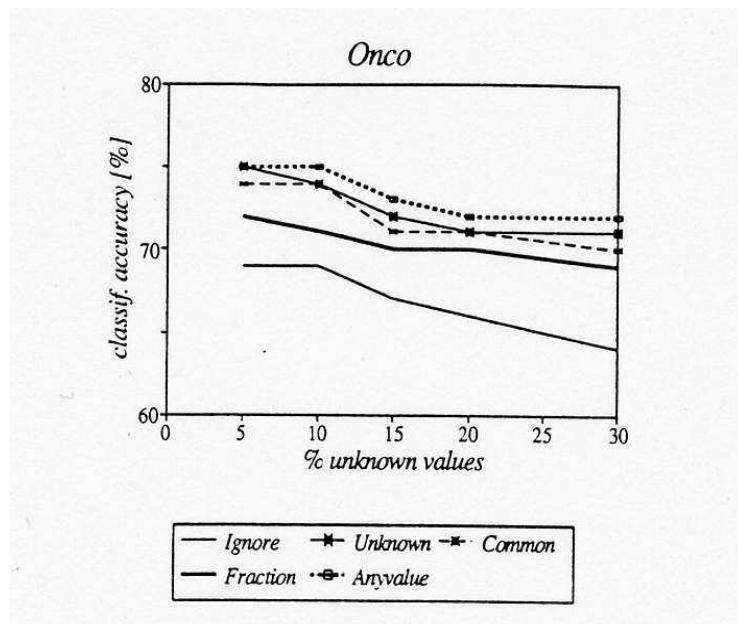
Fig. 2. Classification accuracy vs. percentage of unknown values for the five routines, performed on the dataset *Onco*. Each point represents average accuracy of the three learning modes.
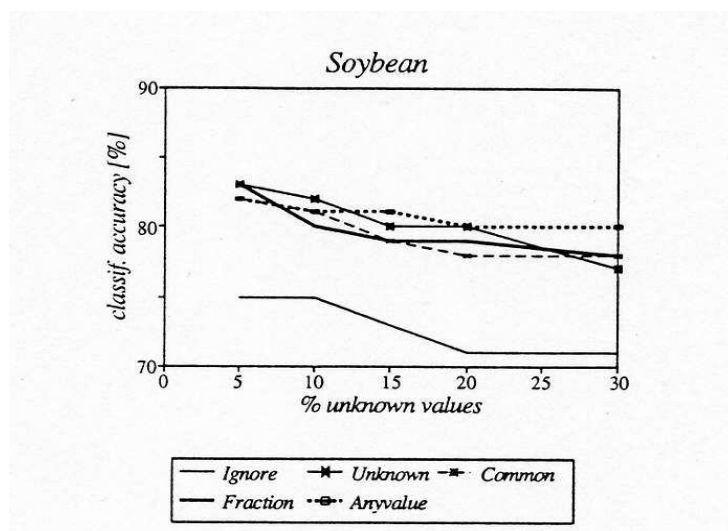

Fig. 3. Classification accuracy vs. percentage of unknown values for the five routines, performed on the dataset *Soybean*. Each point represents average accuracy of the three learning modes.