Experiments with noise filtering in a medical domain

Dragan Gamberger
Rudjer Bošković Institute
Bijenička 54
10000 Zagreb, Croatia
Dragan.Gamberger@irb.hr

Nada Lavrač
Jozef Stefan Institute
Jamova 39
1000 Ljubljana, Slovenia
Nada.Lavrac@ijs.si

Ciril Grošelj
University Medical Centre Ljubljana
Zaloška 7
1000 Ljubljana, Slovenia

Abstract

The paper presents a series of noise detection experiments in a medical problem of coronary artery disease diagnosis. The following algorithms for noise detection and elimination are tested: a saturation filter, a classification filter, a combined classification-saturation filter, and a consensus saturation filter. The distinguishing feature of the novel consensus saturation filter is its high reliability which is due to the multiple detection of potentially noisy examples. Reliable detection of noisy examples is important for the analysis of patient records in medical databases, as well as for the induction of rules from filtered data, representing genuine characteristics of the diagnostic domain. Medical evaluation in the problem of coronary artery disease diagnosis shows that the detected noisy examples are indeed noisy or non-typical class representatives.

1 INTRODUCTION

Effective noise handling is one of the most difficult problems in inductive machine learning. The prediction accuracy and applicability of induced rules significantly depend on appropriate noise handling procedures. Noise usually means random errors in training examples (erroneous attribute values and/or erroneous classification). In this work we use the term noise in a broader sense of outliers [17] denoting all examples that do not follow the same model as the rest of the data. This definition "includes not only erroneous data but also *surprising* veridical data" [10], including non-typical class representatives.

A *target concept* of a given problem domain is defined as the source of all possible correct examples. An inductive learning task is to find a good representation of the target concept in a selected hypothesis language. This representation is called a *target theory*. The main property of a target theory is that it should be correct for all the correct domain examples. A domain may consist of instances of a single concept, a set of subconcepts, a main concept and some subconcepts, etc. These domain characteristics together with the restrictions of the used hypothesis language may cause that in some cases a target theory may have the form of a single theory description, a set of subtheory descriptions or a main theory description and its exceptions [6].

In an ideal inductive learning problem, the induced hypothesis H is indeed a target theory that will 'agree' with the classifications of training examples E and will thus perform as a perfect classifier on yet unseen instances. In practice, however, it frequently happens that data given to the learner contain various kinds of errors, either random or systematic. Random errors are usually referred to as *noise*. Therefore, in most real-life problems the success of machine learning very much depends on the learner's noise-handling capability, i.e., its ability of appropriately dealing with noisy data. Although noise in training examples may be due to erroneous attribute values and erroneous class labels, machine learning algorithms usually treat noisy examples as being mislabeled.

It should be noted again that the term noise used in this work does not refer only to errors in the data. As opposed to the standard terminology, the term noise is here used as a synonym for outliers: it refers to errors (incorrect examples) as well as exceptions (correct examples representing some relatively rare subconcept of the target concept). The reasoning behind this deci-

sion is that, from the point of view of induction, exceptions have the same effect on the induction process as erroneous examples themselves. Distinguishing exceptions from noise and hypothesis generation including the representation of exceptions have been studied also by other authors, for example [16, 3].

Detection and elimination of noisy examples from the training set helps in the induction of the target hypothesis - a hypothesis induced from noiseless data will be less complex and more accurate when classifying unseen cases. In contrast to the so-called noise tolerant procedures for noise handling, such as rule truncation and tree pruning [14, 15], this work is concerned with the explicit detection and elimination of noisy data in data preprocessing. This work upgrades the earlier work of the authors on Occam's razor applicability for noise detection and elimination [7, 8]. The approach is based on the observation that the elimination of noisy examples, in contrast to the elimination of examples for which the target theory is correct, reduces the CLCH value of the training set (CLCH stands for the Complexity of the Least Complex correct Hypothesis). This noise detection algorithm is called the *saturation filter*, since it employs the CLCH measure to test whether the training set is saturated, i.e., whether it can be used to induce a stable target theory. Recently, an alternative approach to explicit noise detection and elimination has been suggested [1]. The basic idea of this noise elimination algorithm, called a *classification filter* in this paper, is to use one or more learning algorithms (that may but do not have to include explicit noise handling) to create classifiers that serve as filters for the training data. The experimental results published in [1] provide evidence that classification filters can successfully deal with noisy data.

In this work, a problem of diagnosis of coronary artery disease [9] is used to perform a series of noise detection experiments. Included are the experiments with our rule learner ILLM (Inductive Learning by Logic Minimization) [5] used without its noise handling capability, the saturation filter used with ILLM, the classification filter with ILLM as the incorporated classifier, and two novel approaches: a combination of noise filtering approaches called the *combined classification-saturation filter* and the *consensus saturation filter* with various consensus levels which employ different ways of n-fold saturation filtering, aimed at increasing the reliably of noisy example detection.

2 NOISE FILTERING ALGORITHMS

Four noise filtering algorithms are presented, aimed at the detection and elimination of noisy training examples: our noise filtering algorithm (here called the saturation filter, [7, 8]), the noise filtering algorithm by Brodley and Friedl (here called the classification filter, [1]) and two new algorithms that combine the two approaches to noise filtering.

2.1 SATURATION FILTER

Our approach to noise filtering has its theoretical foundation in the saturation property of training data [8]. The algorithm, described in detail in [7], is outlined here for the sake of completeness.

Suppose that a complexity measure c is defined and that for any hypothesis H its complexity $c(H)$ can be determined. Based on this complexity measure, for a training set E one can determine the complexity of the least complex hypothesis correct for all the examples in E; this complexity, denoted by $g(E)$ is called the CLCH value (Complexity of the Least Complex Hypothesis, correct for all the examples in E).

In [8] we have shown that if E is noiseless and saturated (containing enough training examples to find a correct target hypothesis), then $g(E) < g(E_n)$, where $E_n = E \cup \{e_n\}$ and e_n is a noisy example for which the target hypothesis is not correct. The property $g(E) < g(E_n)$ means that noisy examples can be detected as those that enable CLCH value reduction. The approach in an iterative form is applicable also when E_n includes more than one noisy example.

It must be noted that the saturation property of a training set is the main theoretical condition for the presented filter. In practice many domains have a restricted number of training examples and hence we may assume that these domains do not satisfy the saturation condition. Notice, however, that the described algorithm is applicable without changes also in this case. The reason is that for some subconcept of the domain there may still be enough training examples so that this subpart of the domain is saturated; hence, a subtheory description is induced by the learner whereas all other examples are eliminated since the learner will treat them as being erroneous or expections of the subtheory description being learned [6].

The greatest practical problem of the saturation fil-

ter is the computation of the CLCH value $g(E)$ for a training set E. In rule-based induction, the hypothesis complexity measure $c(H)$ can be defined as the number of attribute value tests (literals) used in the hypothesis H. In this case, the corresponding $g(E)$ value can be defined as the minimal number of literals that are necessary to build a hypothesis that is correct for all the examples in E.

Suppose that the training set is contradiction free (there are no examples that differ only in their class value), and that the set of literals L defined in the hypothesis language is sufficient for finding a hypothesis H which is correct for all examples in E. Then the necessary and sufficient condition for a subset $L' \subseteq L$ to have this same property is that for every possible example pair, such that the first example in the pair is a positive example from E and the second one is negative, there must be at least one literal in L' which covers the pair. A literal covers a pair if it is true for the positive and false for the negative example in the pair. This fact enables that the $g(E)$ value defined by the minimal number of literals can be computed by any minimal covering algorithm over the set of example pairs. In this work, the ILLM heuristic minimal covering algorithm, presented here as Procedure 1 in Figure 1, is used. The advantages of this approach are: $g(E)$ computation does not require the actual construction of a hypothesis and the $g(E)$ value can be determined relatively fast. This approach presents the heart of the saturation noise filtering method used in this work.

The procedure starts with the empty set of selected literals L' (step 1) and the set U' of yet uncovered example pairs equal to all possible pairs of one positive and one negative example from the training set (step 3), for which in (step 2) weights $v(e_i, e_j)$ have been computed. The weight of a pair is high if the pair is covered by a small number of distinct literals from L. The meaning of this measure is that for a pair with a high weight it will be more difficult to find an appropriate literal which will cover this pair than for a pair with a small weight.

Each iteration of the main algorithm loop (steps 4 – 11) adds one literal to the minimal set L' (step 9). At the same time, all example pairs covered by the selected literal are eliminated from U' (step 10). The algorithm terminates when U' remains empty. In each iteration we try to select the literal which covers a maximal number of 'heavy' example pairs (pairs with high weight). This is achieved so that a pair (e_a, e_b) is detected which is covered by the least number of

Procedure 1: MINIMAL COVER
Input: U (set of example pairs), L (set of literals)
Output: L' (minimal set of literals)
(1) $L' \leftarrow \emptyset$
(2) **for** every $(e_i, e_j) \in U$ compute weights
    $v(e_i, e_j) = 1/z$, where z is the number of literals $l \in L$ that cover (e_i, e_j)
(3) $U' \leftarrow U$
(4) **while** $U' \neq \emptyset$ **do**
(5) select (e_a, e_b)
    $(e_a, e_b) \in U'$: $(e_a, e_b) = arg\ max\ v(e_i, e_j)$,
    where max is over all $(e_i, e_j) \in U'$
(6) $L_{ab} \leftarrow \{l \mid l \in L$ covering $(e_a, e_b)\}$
(7) **for** every $l \in L_{ab}$ compute
    $w(l) = \sum v(e_i, e_j)$, where sum is over all $(e_i, e_j) \in U'$ covered by l
(8) select literal l_s: $l_s = arg\ max\ w(l)$,
    where max is over all $l \in L_{ab}$
(9) $L' \leftarrow L' \cup \{l_s\}$
(10) $U' \leftarrow U' \setminus \{$all (e_i, e_j) covered by $l_s\}$
(11) **end while**

Figure 1: Heuristic minimal covering algorithm.

literals (step 5). At least one of the literals from the set L_{ab} with literals that cover this pair (step 6), must be included into the minimal set L'. To determine this literal, for each of them weight $w(l)$ is computed (step 7) and the literal with the maximal weight is selected (step 8). The weight of a literal is the sum of the weights of example pairs that are covered by the literal.

Algorithm 1 in Figure 2 presents the saturation filter. It begins with the reduced training set E' equal the input training set E (step 1) and an empty set of detected noisy examples A (step 2). The algorithm supposes that the set of all appropriate literals L for the domain is defined. U represents a set of all possible example pairs where the first example in the pair is from the set of all positive training examples P' in the reduced set E', and the second example is from the set N' of all negative examples in the reduced training set E'. The algorithm detects one noisy example per iteration. The base for noise detection are weights $w(e)$ which are computed for each example e from E'. Initially all $w(e)$ values are initialized to 0 (step 6). At the end, the example with maximum weight $w(e)$ is selected (step 17). If the maximum $w(e)$ value is greater than the parameter ε_h predefined value then the corresponding training example is included into the set A

Algorithm 1: $SaturationFilter(E)$
Input: E (training set), L (set of literals)
Parameter: ε_h (noise sensitivity parameter)
Output: A (detected noisy subset of E)

(1) $E' \leftarrow E$
(2) $A \leftarrow \emptyset$
(3) **while** $E' \neq \emptyset$ **do**
(4) find U, set of all possible example pairs for examples in E'
(5) call **Procedure 1** to find minimal L', $L' \subseteq L$ so that $\forall (e_i, e_j) \in U \ \exists \ l \in L'$ with the property l covers (e_i, e_j)
(6) initialize $w(e) \leftarrow 0$ for all $e \in E'$
(7) **for** every $l \in L'$ **do**
(8) $P^* \leftarrow \emptyset$, $N^* \leftarrow \emptyset$
(9) **for** every $(e_i, e_j) \in U$
(10) **if** (e_i, e_j) covered by l and no other literal from L' **then**
(11) $P^* \leftarrow P^* \cup \{e_i\}$, $N^* \leftarrow N^* \cup \{e_j\}$
(12) **end for**
(13) **if** $P^* = \emptyset$ **then** $L' \leftarrow L' \setminus \{l\}$ and **goto** step 6
(14) **for** every $e \in P^*$ **do** $w(e) \leftarrow w(e) + \frac{1}{|P^*|}$
(15) **for** every $e \in N^*$ **do** $w(e) \leftarrow w(e) + \frac{1}{|N^*|}$
(16) **end for**
(17) select example e_s: $e_s = arg\ max\ w(e)$, where max is computed over all $e \in E'$
(18) **if** $w(e_s) > \varepsilon_h$ **then**
(19) $A \leftarrow A \cup \{e_s\}$
(20) $E' \leftarrow E' \setminus \{e_s\}$
(21) **else** exit with generated sets A and E'
(22) **end while**

Figure 2: Saturation filter.

(step 19) and eliminated from the reduced training set E' (step 20). The new iteration of noise detection begins with this reduced training set (steps 3–22). The algorithm terminates when in the last iteration no example has $w(e)$ greater than ε_h. Noisy examples in A and the noiseless E' are the output of the algorithm.

Computations in each iteration begin with the search for the minimal set of literals L' that cover all example pairs in U (calling Procedure 1 in step 5). A pair of examples is covered by a literal l if the literal is evaluated *true* for the positive example and evaluated *false* for the negative example in the pair. This step represents the computation of the $g(E')$ value. Next, a heuristic approach is used to compute weights $w(e)$ that measure the possibility that the elimination of an example e would enable $g(E')$ reduction. Weights $w(e)$ are computed so that for every literal l from L', minimal sets of positive (P^*) and negative examples (N^*) are determined, such that if P^* or N^* are eliminated from E' then l becomes unnecessary in L'. This is done in a loop (steps 9–12) in which every example pair is tested if it is covered by a single literal l. If such a pair is detected (step 10) then its positive example is included into the set P^* and its negative example into the set N^* (step 11). Literal elimination from L' presents the reduction of the $g(E')$ value. If a literal can be made unnecessary by the elimination of a very small subset of training examples, then this indicates that these examples might be noisy. In steps 14 and 15, the $w(e)$ weights are incremented only for the examples which are the members of the P^* and N^* sets. The weights are incremented by the inverse of the total number of examples in these sets. Weights are summed over all literals in L'. Step 13 is necessary because of the imperfectness of the heuristic minimal cover algorithm. Namely, if some $l \in L'$ exists for which there is no example pair that is covered only by this literal (i.e., for which either $P^* = \emptyset$ or $N^* = \emptyset$), this means that L' is actually not the minimal set because $L' \setminus \{l\}$ also covers all example pairs in U. In such case L' is substituted by $L' \setminus \{l\}$.

The presented saturation filter uses the parameter ε_h that determines noise sensitivity of the algorithm. The parameter can be adjusted by the user in order to tune the algorithm to the domain characteristics. Reasonable values are between 0.25 and 2. For instance, the value 1.0 guarantees the elimination of every such example by whose elimination the set L' will be reduced for at least one literal. Lower ε_h values mean greater sensitivity of the algorithm (i.e., elimination of more examples): lower ε_h values should be used when the domain noise is not completely random, and when dealing with large training sets (since statistical properties of noise distribution in large training sets can have similar effects). In ILLM the default values of ε_h are between 0.5 and 1.5, depending on the number of training examples in the smaller of the two subsets of E: the set of positive examples P or the set of negative examples N. Default values for the saturation filter's noise sensitivity parameter ε_h are: 1.5 for training sets with 2–50 examples, 1.0 for 51–100 examples, 0.75 for 101–200 examples, and 0.5 for more than 200 examples.

Algorithm 2: $ClassificationFilter(E)$
Input: E (training set)
Parameter: n (number of subsets, typically 10)
Output: A (detected noisy subset of E)
(1) form n disjoint almost equally sized subsets E_i, where $\cup_i E_i = E$
(2) $A \leftarrow \emptyset$
(3) **for** $i = 1, \ldots, n$ **do**
(4) form $E_y \leftarrow E \setminus E_i$
(5) induce H_y based on examples in E_y (using some inductive learning system)
(6) **for** every $e \in E_i$ **do**
(7) **if** H_y incorrectly classifies e **then** $A \leftarrow A \cup \{e\}$
(8) **end for**
(9) **end for**

Figure 3: Classification filter.

Algorithm 3: $ConsensusSaturationFilter(E)$
Input: E (training set)
Parameter: n (number of subsets, typically 10)
Parameter: v (consensus level, typically $n-1$)
Output: A (detected noisy subset of E)
(1) form n disjoint almost equally sized subsets E_i, where $\cup_i E_i = E$
(2) $A \leftarrow \emptyset$
(3) **for** $i = 1, \ldots, n$ **do**
(4) form $E_y \leftarrow E \setminus E_i$
(5) $A_i \leftarrow SaturationFilter(E_y)$
(6) **end for**
(7) **for** every $e \in E$ **do**
(8) $c \leftarrow 0$
(9) **for** $i = 1, \ldots, n$ **if** $e \in A_i$, $c \leftarrow c + 1$
(10) **if** $c \geq v$, $A \leftarrow A \cup \{e\}$
(11) **end for**

Figure 4: Consensus saturation filter.

2.2 CLASSIFICATION FILTER

The n-fold cross-validation method is a substantial part of this filtering algorithm. The classification filter (Algorithm 2, shown in Figure 3) begins with n equal-sized disjoint subsets of the training set E (step 1) and the empty output set A of detected noisy examples (step 2). The main loop (steps 3–9) is repeated for each training subset E_i. In step 4, subset E_y is formed which includes all examples from E except those in E_i. Set E_y is used as the input for an arbitrary inductive learning algorithm that induces a hypothesis (a classifier) H_y (step 5). Those examples from E_i for which the hypothesis H_y does not give the correct classification are added to A as potentially noisy examples (step 7).

2.3 COMBINED CLASSIFICATION-SATURATION FILTER

A combined classification-saturation algorithm is very similar to the original classification filtering approach. The only difference is that saturation filtering is used for every subset E_y in order to eliminate noise from E_y. The intention of this modification is the induction of more appropriate hypotheses H_y and more reliable noise detection based on these. The combined classification-saturation filter algorithm is the same as the classification filter with added saturation-based filtering between its steps 4 and 5.

2.4 CONSENSUS SATURATION FILTER

The n-fold cross-validation method is also a substantial part of the consensus saturation filtering algorithm (Algorithm 3, shown in Figure 4). Like in the combined classification-saturation filter, subset E_y is constructed (step 4) and noise eliminated from it by a saturation filter. In contrast to the combined classification-saturation filter, this reduced E_y is not used for hypothesis induction. Instead, detected noise is simply saved in the set A_i (step 5). When the procedure is performed for all E_i subsets, then n generated A_i sets are used to determine the elements of the output set A (steps 7–11). Since the same example e occurs in $n-1$ of E_y subsets, in an ideal case the same noisy example may occur in $n-1$ sets A_i. In this situation, detected by counter c (steps 9–11), example e is added to the output noisy set A (step 10), since the consensus of all A_i sets is reached. In practice we may allow that an example is detected as noisy and added to A also in cases when it is included in less than $n-1$ sets A_i. Parameter v with values less than $n-1$(e.g., $n-2$, $n-3$) enables this possibility.

3 EXPERIMENTAL EVALUATION

3.1 DOMAIN DESCRIPTION: DIAGNOSIS OF CORONARY ARTERY DISEASE

Coronary artery disease (CAD) is a result of diminished blood flow through coronary arteries due to stenosis or occlusion. The consequence of CAD is an impaired function of the heart and possible necrosis of the myocardium (myocardial infarction).

The dataset, collected at the University Medical Center, Ljubljana, Slovenia, includes 327 patients (250 men and 77 women, mean age 55 years). Each patient had performed clinical and laboratory examinations including ECG during rest and exercise, myocardial perfusion scintigraphy and finally the coronary angiography that provides for the actual diagnosis of coronary artery disease. In 229 patients, CAD was confirmed by angiography, and for 98 patients it was not confirmed. The patients' clinical and laboratory data are described by 77 attributes. This dataset was previously used for inducing diagnostic rules by a number of machine learning algorithms [9, 11].

3.2 EXPERIMENTAL DESIGN AND RESULTS

According to the standard 10-fold cross-validation procedure, the original data set was partitioned into 10 folds with 32 or 33 examples each. Training sets are built from 9 folds, leaving one fold is a test set. Let G denote the entire set of training examples, T_i is an individual test set (consisting of one fold), and G_i the corresponding training set ($G_i \leftarrow G \setminus T_i$, composed on nine folds). In this way, 10 training sets G_0 - G_9, and 10 corresponding test sets, T_0 - T_9, were constructed. Every example occurs exactly once in a test set, and 9 times in training sets.

Different noise detection procedures were used on the training sets and after the elimination of potentially noisy examples and hypothesis generation the prediction accuracy was measured on the test sets. The hypothesis was always constructed with the same algorithm so that the differences in the obtained prediction accuracy reflect only the differences in noise detection. The used rule construction algorithm was the ILLM (Inductive Learning by Logic Minimization) system used without its noise handling capability. The ILLM rule learning algorithm is similar to the AQ15 and CN2 covering algorithms for rule construction [2, 4].

In the first test we used ILLM (without its noise handling capability) to induce rules on complete G_0 - G_9 training sets. On the test sets T_0 - T_9 we measured the number of prediction errors: on the average, there were 5.5 prediction errors per test set. This corresponds to the 83.1% average prediction accuracy. This result, presented in the first row (A) of Table 1, presents the baseline for comparing the quality of different noise elimination algorithms. The average number of eliminated training examples is 0 because no noise elimination has been used. Generated rules are complete and consistent with all training examples in corresponding training sets.

In all other experiments instead of G_0 - G_9, reduced training sets G'_0 - G'_9 were used as the input to inductive learning. Reduced training sets G'_i were obtained by the elimination of noisy examples, $G'_i \leftarrow G_i \setminus A$, where A represent noisy set outputs of different filtering algorithms for G_i as input training sets.

3.2.1 Saturation Filter

By using the saturation filter presented in Section 2.1 with the ε_h parameter set to 1.0, the average prediction error was 3.6 examples per test set, which corresponds to the 89.0% average prediction accuracy. The result is presented in the second row (S) of Table 1. Compared to the results in row A this is a substantial improvement. This result was obtained by an average elimination of 30.2 examples per iteration, which represents about 10% of the training sets. A comparison to prediction results obtained by other authors, using very different machine learning methods, which are all between 86.6% and 89.7% [11], indicates that 10% of detected and eliminated noise seems to be realistic. Both the prediction accuracy and the number of eliminated examples demonstrate that saturation filtering can be used as an effective noise handling mechanism.

3.2.2 Classification Filter

Using the classification filter, described in Section 2.2, with parameter $n = 10$ for noise detection, in average 4.7 prediction errors per test set were made. This corresponds to the 85.6% average prediction accuracy. This result, presented in the third row (C) of Table 1, represents an improvement in accuracy when compared to the result of row A although it is not as good as the result obtained by the saturation filter (row S). Nevertheless, this result is important because the classification filter is computationally much simpler than the saturation filter. Additionally, according to [1], the use of simple voting mechanisms based on the results of filtering by different learning approaches (not neces-

Algorithm	Predict. err.		Accuracy	Elimin. ex.	
A	5.5	(3.03)	83.1 %	0	
S	3.6	(1.96)	89.0 %	30.2	(2.78)
C	4.7	(1.64)	85.6 %	52.6	(4.58)
CS	3.6	(1.71)	89.0 %	36.5	(4.34)
S9	4.2	(1.48)	87.2 %	12.5	(2.46)
S8	3.6	(1.17)	89.0 %	17.5	(2.12)
S7	3.6	(1.35)	89.0 %	19.8	(2.20)

Table 1: Average number of prediction errors (with standard deviation in parentheses), average prediction accuracy, and average number of eliminated examples (with standard deviation in parentheses) for different noise detection algorithms: A - ILLM without noise handling, S - saturation filter, C - classification filter, CS - combined classification-saturation filter, S9,S8,S7 - consensus saturation filters with varied levels of consensus.

sarily only inductive learning approaches) can further improve the reliability of this noise detection process. This possibility was not investigated in this work.

The average number of eliminated examples per training set was 52.6. This means that the classification filter practically detected every fifth training example as potentially noisy. This shows a weakness of the classification filter: it is non-selective. Too many examples are detected as being potentially noisy.

3.2.3 Combined Classification-Saturation Filter

Results obtained with noise filtering performed by the combined classification-saturation filter (see Section 2.3, n=10, $\varepsilon_h = 1.0$) are presented in CS row of Table 1. The measured average error was 3.6 examples representing the 89.0% average accuracy. It can be concluded that the combination of the two different noise detection algorithms is advantageous only in comparison with the classification filtering approach. The combined approach namely resulted in the average of 36.5 detected and eliminated potentially noisy examples per training set. Although this number is smaller than the one for the classification filter (row C with 52.6 examples) it seems that non-selectiveness is an inherent characteristic of classification filtering when a single classifier is used.

3.2.4 Consensus Saturation Filter

The relative success of the classification filter, which enables significant prediction accuracy improvement using a learning algorithm without noise handling, stimulated a series of experiments in which we tried to test if a mechanism similar to the one used in the classification filter could improve the results of saturation filtering. In all the experiments parameter n was set to 10 and ε_h=1.0, while changing the consensus level v. With a default value $v = 9$ the average number of eliminated examples per training set was only 12.5 and the average number of prediction errors on test sets was 4.2 (presented in S9 row of Table 1).

This result is important since a satisfactory prediction accuracy was achieved by the elimination of a very small number of training examples. When compared with saturation filtering itself (row S) we see that by requiring the consensus of filtering in 10-fold validation, lower prediction accuracy (87.2% instead of 89.0%) was achieved by a significantly smaller average number of eliminated examples (12.5 instead of 30.2).

The described consensus filtering approach seems to be a reliable noise detection algorithm. A further proof of this claim is the following: from the total of 125 examples eliminated by the consensus filter (row S9), 124 of them were detected also by the combined classification-saturation filter (row CS) which, however, eliminated 365 examples in total.

Requiring consensus in 10-fold validation results in noise detection of high specificity. However, the decreased prediction accuracy indicates that the algorithm's sensitivity is too low. This can be the consequence of a too high consensus level (in order to declare an example to be potentially noisy, the example had to be tagged as potentially noisy in all subsets in which it occurs, i.e., in 9 subsets).

The algorithm's noise sensitivity can easily be increased by decreasing the consensus level v. In this way, relaxed consensus filters can be constructed. Rows S8 and S7 present results obtained by setting the consensus levels to 8 and 7, respectively. In both cases the increase in sensitivity resulted in the increase of the achieved prediction accuracy. Good prediction accuracy of 89.0%, as in cases S and CS, is obtained by relaxed consensus filtering but with significantly fewer potentially noisy examples eliminated.

3.3 MEDICAL EVALUATION

The results of described experiments suggest that consensus saturation filtering (with consensus level 9, row S9 of Table 1) presents a reliable tool for the detection of examples that are indeed noisy. In order to test the practical usefulness of noise detection by the consensus saturation filter, we applied the described approach to

the whole coronary artery disease dataset of 327 examples. In this case the training sets G_0 - G_9 were used as 10 training subsets for the consensus filter. In total 15 potentially noisy examples were detected. This is in accordance with the result obtained for G_0 - G_9 sets for which the average value of eliminated examples was 12.5.

The detected examples were shown to a domain expert for evaluation. In order to make the task more difficult to the expert, we formed a set of 20 examples consisting of 15 examples detected as noisy by the consensus saturation filter, and additional 5 randomly selected non-noisy examples. The expert analyzed the group of these 20 examples as a whole. Out of these 20 examples, 13 were of class non-confirmed (11 detected potentially noisy and 2 other examples of class non-confirmed), and 7 were of class confirmed (4 detected potentially noisy examples and 3 other examples of class confirmed).

In the class non-confirmed the expert recognized 11 cases as being outliers, and these examples were exactly those selected by the noise detection algorithm. For 8 of these 11 cases the problem was a high grade of stenosis of coronary arteries, very close to the predefined value that distinguishes between patients with confirmed and non-confirmed coronary artery disease. Other 3 cases represented patients who did not have actual coronary heart disease problems but rather problems due to recent miocardial infarction, functionally malfunctioning by-pass, and valve disease.

For the class confirmed the expert detected 4 potentially noisy cases three of which were selected also by the noise detection algorithm and one of the randomly selected cases of this class. For all 4 cases the detected grade of stenosis was very close to the border line between the two classes. Additionally, for one patient the values of measured parameters were detected as non-typical due to the improper level of stress during measurements.

The expert has also analyzed four diagnostic parameters MAIN (main coronary artery), LAD (left anterior descendens), LCX (left circumflex) and RCA (right coronary artery). Each of these parameters has values between 0 and 3 and denotes the level of stenosis or occlusion in the corresponding artery. The value 0 denotes no stenosis, value 1 up to 50%, value 2 corresponds to 50%–75%, and value 3 represents more than 75% of stenosis or occlusion. Average values of these parameters are computed for all non-confirmed class cases and 11 non-confirmed class cases selected as noise, as well as for all confirmed class cases and 4 confirmed class cases selected as noise. Results are presented in Table 2. It can be noticed that for all four parameters, noisy non-confirmed cases have higher average values than those computed for the whole class. On the contrary, for the confirmed class, for all four parameters noisy examples have lower average values than average values for the whole class of 229 cases.

	non-confirmed cases		confirmed cases	
	all	11 noisy	all	4 noisy
MAIN	0.01	0.09	1.53	0.50
LAD	0.29	0.73	2.15	0.75
LCX	0.10	0.45	1.31	0
RCA	0.15	0.18	1.38	0

Table 2: Average values for 4 diagnostic parameters.

The analysis shows that, except for a few patients who did not have actual coronary heart disease problems, most of cases detected as noise represent a special group of patients whose coronary angiography results are very close to the border line between the two classes CAD confirmed and not-confirmed. By a slight change of the definition of class confirmed these cases can change their class value, therefore it is clear that they are non-typical training cases. Their elimination from the training set, as well as the elimination of patients who did not have actual coronary heart disease problems and patients with improperly measured parameters, is reasonable if one wants to induce diagnostic rules uncovering characteristic properties of patients with CAD.

4 CONCLUSION

Explicit noise detection and elimination turns out to be a very useful and important step in data preprocessing for any inductive learning algorithm. Namely, the detection and filtering of potentially noisy examples is itself an important result that may be useful in practice for data cleansing.

Standard deviations, presented as parenthesized numbers in Table 1, show that the differences of achieved prediction accuracies by different noise filtering algorithms are not significant. On the other hand, most of the differences in the numbers of eliminated noisy examples are significant and represent genuine characteristics of the presented noise filtering algorithms.

It is interesting that a simple classification filtering approach, based on an iterative application of a ma-

chine learning algorithm (without noise handling) by itself enables an improvement of the prediction accuracy. However, in the coronary artery disease diagnostic domain, best prediction accuracies were achieved by the approaches which include saturation filtering. The proposed consensus saturation filter is an interesting combination: it eliminates only a small number of potentially noisy examples, with a very high probability of actually being noisy. This property may turn out to be decisive for a broader applicability of noise detection algorithms. The experiments suggest that a relaxed consensus saturation filter (see row S8 in Table 1) represents a very good solution to the problem of noise filtering.

Acknowledgement

We are grateful to Matjaž Kukar from the Faculty of Computer and Information Sciences, University of Ljubljana, for his help and involvement in the evaluation of the results of this study.

References

[1] Brodley, C.E. & Friedl, M.A. (1996). Identifying and eliminating mislabeled training instances. In *Proc. of the 13th National Conference on Artificial Intelligence*, 799–805. AAAI Press.

[2] Clark, P. & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning* **3**: 261–283.

[3] Dimopoulos, Y. and Kakas, A (1995) Learning non-monotonic logic programs: Learning exceptions. In N. Lavrač and S. Wrobel, editors, Proc. of the 8th European Conference on Machine Learning, volume 912 of Lecture Notes in Artificial Intelligence, 122–137. Springer-Verlag, 1995.

[4] Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review* **13**(1): 3–54.

[5] Gamberger, D. (1995). A minimization approach to propositional inductive learning. In *Proc. of the 8th European Conference on Machine Learning*, 151–160. Springer, Berlin.

[6] Gamberger, D. & Lavrač, N. (1996). Noise detection and elimination applied to noise handling in a KRK chess endgame. In *Proc. of the 6th International Workshop on Inductive Logic Programming*, 59–75. Springer, Berlin.

[7] Gamberger, D., Lavrač, N. & Džeroski, S. (1996). Noise elimination in inductive concept learning: A case study in medical diagnosis. In *Proc. of the 7th International Workshop on Algorithmic Learning Theory*, 199–212. Springer, Berlin.

[8] Gamberger, D. & Lavrač, N. (1997). Conditions for Occam's razor applicability and noise elimination. In *Proc. of the 9th European Conference on Machine Learning*, 108–123. Springer, Berlin.

[9] Grošelj, C., Kukar, M., Fetich, J. & Kononenko, I. (1997). Machine learning improves the accuracy of coronary artery disease diagnostic methods. *Computers in Cardiology* **24**: 57–60.

[10] John, G.H. (1995). Robust decision trees: Removing outliers from data. In *Proc. of the 1st Int. Conference on Knowledge Discovery and Data Mining*, 174–179. AAI Press.

[11] Kukar, M., Grošelj, C., Kononenko, I. & Fetich, J. (1997). An application of machine learning in the diagnosis of ischaemic heart disease. In *Proc. of the 6th Conference on Artificial Intelligence in Medicine Europe*, 461–464. Springer, Berlin.

[12] Lavrač, N., Gamberger, D. & Džeroski, S. (1995). An approach to dimensionality reduction in learning from deductive databases. In *Proc. of the 5th International Workshop on Inductive Logic Programming (ILP-95)*, 337–354. Katholieke Universiteit Leuven Scientific Report.

[13] Lavrač, N., Gamberger, D. & Turney, P. (1998). A relevancy filter for constructive induction. *IEEE Intelligent Systems* **13**: 50–56.

[14] Mingers, J. (1989). An empirical comparison of pruning methods for decision tree induction. *Machine Learning* **4**(2): 227–243.

[15] Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

[16] Srinivasan, A., Muggleton, S. & Bain, M. (1992). Distinguishing exceptions from noise in non-monotonic learning. In *Proc. of the 2nd Int. Workshop on Inductive Logic Programming*. Tokyo, ICOT TM-1182.

[17] Weisberg, S. (1985). *Applied Linear Regression*. John Wiley & Sons.