Correcting Noisy Data

Choh Man Teng
cmteng@cse.unsw.edu.au

School of Computer Science and Engineering
University of New South Wales
Sydney NSW 2052 Australia

Abstract

Inductive learning aims at constructing a generalized description of a given set of data, so that future similar instances can be classified correctly. The performance on this task depends crucially on the quality of the data. We investigate here an approach to handling noise in the training data by identifying possible noisy attributes and/or class in each instance, and replacing such values with more appropriate ones. The resulting data set would preserve much of the original information, but conform more to the ideal noise-free case. A classifier built from this corrected data should have a higher predictive power. We make use of the interdependence among attributes and between the attributes and the class to predict the value of one attribute using the rest of the attributes together with the class value. These predictions serve as candidates for possible adjustments to a training instance that has been misclassified. We selectively adjust some of the attribute values accordingly to obtain a better fit of the polished instance to the classifier. Preliminary experimentation suggests that this is a viable approach to noise reduction and correction.

1 INTRODUCTION

Inductive learning aims at constructing a generalized description of a given set of data, so that future similar instances can be classified correctly. The performance of this task depends crucially on the quality of the data. When noise is present, the classifier built from this data would be less accurate a description of the target concept and thus be of a lower utility. Unfortunately it is inevitable that noise exists in real world data sets, due to clerical error, faulty measuring devices, and the like. Thus, noise handling is a major issue in machine learning systems.

The standard approach to coping with noise is to avoid overfitting, so that the classifier does not build overly complicated structures just to fit the noise [Quinlan, 1987; Clark and Niblett, 1989]. These classifiers are robust and the noisy instances are retained as part of the data set. Another approach tackles the problem from the input end, and eliminates from the data set instances that are suspect of noise according to certain evaluation mechanisms [John, 1995; Brodley and Friedl, 1996; Gamberger *et al.*, 1996]. A classifier is then built using only the retained instances. Similar ideas can be found in robust regression and outlier detection techniques in statistics [Rousseeuw and Leroy, 1987].

In removing noisy instances from the data, there is a tradeoff between the amount of information available for building the classifier and the amount of noise retained in the data set. We investigate a third approach, namely, correcting the noisy instances rather than eliminating them. The resulting data set would preserve much of the original information, but conform more to the ideal noise-free case. A classifier built from this corrected data should have a higher predictive power.

The rest of the paper is organized as follows. The basic methodology of the noise correction procedure we call *polishing* is described in Section 2. In Section 3 we outline the setup for experimental evaluation, and the preliminary results are reported in Sections 4 and 5. Additional discussion follows in Section 6, and we conclude in Section 7.

2 BASIC METHODOLOGY

The naive Bayes classifier assumes that the attributes are conditionally independent given the target class. It has often been pointed out that this assumption is a gross over-simplification of the actual relationship between the attributes; hence the word "naive" [Mitchell, 1997, for example]. Extensions to the naive Bayes classifier have been introduced to loosen the independence criterion [Kononenko, 1991; Langley *et al.*, 1992], but some have also investigated alternative explanations for the success of this classifier [Domingos and Pazzani, 1996].

Controversy aside, most would agree attribute dependence is exhibited in a wide variety of settings. The idea behind polishing is very simple. Instead of assuming conditional independence, we exploit the interdependence between attributes, as well as between the attributes and the target class. We assume that there is some pattern of relationship among the different components of a data set. Just as we can predict the target concept by examining the attribute values, we can turn the process around and use the target class value together with some attributes to predict the value of another attribute. Note that except for totally irrelevant attributes, each attribute would at least be related to some extent to the target class even if not to another attribute. This idea is related to structured induction [Shapiro, 1987], where classifiers for individual attributes are built from a set of other attributes.

In a noisy instance, some of the attributes and/or class have inappropriate values. We can increase the classification accuracy by polishing the data; that is, by changing some of the incorrect attribute or class values back to their non-noisy values. If the combination of the target class and a subset of the attributes is a good predictor of the value of the remaining attributes, we then have a way of determining which attributes or class are noisy in a particular instance.

This is obviously an idealization of the procedure. In practice, the prediction is far from perfect, and incorporating every predicted change into an instance would probably introduce more errors than it fixes. Thus, we adopt a conservative adjustment strategy: we would like to alter the data as little as possible, while still achieving some level of improvement. In the following sections we will investigate experimentally how we can make polishing work, and to what extent it works. But first let us look at the polishing mechanism more closely.

2.1 PROCEDURE

Given a training set, we try to identify suspect attributes and classes and replace their values according to a polishing procedure. The barebones description of polishing is given in Figure 1. `Polishing` makes use of a procedure `flip` to recursively try out different combinations of attribute changes. The function `classify`($\mathit{Classifiers}, x_j, c$) returns the number of classifiers in the set *Classifiers* which classify the instance x_j as belonging to class c.

Each polishing run consists of two stages. In the *prediction phase*, for each attribute we perform a 10-fold cross validation on the data with this attribute as the target and the original class participating as an attribute in its place. In other words, the attribute to be predicted and the original class swap roles. For each attribute, the data set is partitioned in a way to achieve a similar distribution of the attribute-turned-class in all folds. The list of attribute-value pairs predicted for each training instance serves as the recommended changes we would try out in the next phase.

In the *adjustment phase*, we selectively change some of the attribute/class values of an instance in order to obtain a better fit of the polished instance to the classifier. First we perform a 10-fold cross validation on the given training set. For each of the instances that has been misclassified, we try to tweak the attributes and the class so that the instance can be classified uniformly and correctly by the classifiers generated during the cross validation. The only allowable tweaks are the changes from the list of attribute-value pairs nominated during the prediction phase, plus the class value suggested by the classifiers for this particular instance. The rationale behind this is that the pattern of relationship among the attributes and the class would be preserved to a certain extent even in the face of noise, and thus the predicted values are more likely to be the correct ones.

Note that we do not indiscriminately replace all the attribute values with the predicted. We tweak only those instances that were misclassified in the first place, and for these instances we try to keep the number of changes to a minimum, preferring those attributes with a higher classification accuracy during the prediction phase. As we will see, this has the consequence that even though there may be an increase in classification accuracy, a lot of the noise remains uncorrected.

When no combination of attribute value replacement is capable of correcting the classification, we replace the class value of the instance with the one predicted

```
Polishing(OldData, NewData, votes, changes, cutoff)
Input  OldData:  (possibly) noisy data
       votes:  #classifiers that need to agree
       changes:  max #changes per instance
       cutoff:  size of attribute subset considered
Output NewData:  polished data

   for each attribute a_i
      AttList_i ← ∅;
      tmpData ← swap a_i and class c in OldData;
      10-fold cross-validation of tmpData;
      for each instance x_j misclassified
         new ← value of a_i predicted for x_j;
         AttList_i ← AttList_i ∪ {⟨j, new⟩};
      end
   end

   NewData ← ∅;
   AttSorted ← attributes sorted
               in ascending order of |AttList_i|;
   Classifiers ← classifiers from 10-fold
                 cross-validation of OldData;
   for each instance x_j
      for k from 0 to changes
         adjusted ← flip(j, votes, k, cutoff);
         if adjusted then break;
      end
      if (not adjusted) and (∃class c s.t.
         classify(Classifiers, x_j, c) ≥ votes)
      then
         class of x_j ← c;
         NewData ← NewData ∪ {x_j};
      end
   end

flip(j, votes, k, cutoff)
Input  j:  index of the instance to be adjusted
       votes:  #classifiers that need to agree
       k:  #changes yet to be made
       cutoff:  size of attribute subset considered
Output true/false: whether a change has been made
       (also modifies NewData)

   if k = 0 then
      if classify(Classifiers, x_j, class of x_j) ≥ votes
      then
         NewData ← NewData ∪ {x_j};
         return true;
      end
      else return false;
   else for i from 0 to cutoff
      a_i' ← AttSorted[i];
      if ⟨j, new⟩ ∈ AttList_i' then
         attribute a_i' of x_j ← new;
         adjusted ← flip(j, votes, k − 1, i − 1);
         if adjusted then return true;
         reset a_i' of x_j;
      end
   end
   return false;
```

Figure 1: Polishing

by the largest number of classifiers if there is enough agreement between them. The assumption here is that a wrong replacement in the attributes is less harmful than a wrong replacement in the target class, since there are usually a lot more attributes than class, and the effect of assigning an individual attribute a wrong value might not always be noticeable, while a wrong class assignment is definitely counter-productive.

2.2 ADJUSTMENT HEURISTICS

In the prediction phase we collect all the nominated changes to the attributes of each instance, and in the adjustment phase we selectively replace some of the attribute/class values with the suggested ones. There are several considerations we need to bear in mind when trying to correct the data. While the suggested values from the prediction phase might be the correct ones to use, they can also be spurious and the use of these values can instead increase the amount of noise in the data set even further. This is especially true for attributes which, through no fault of their own, bear little relationship to the others and thus the prediction of their values is of poor accuracy.

One way to alleviate this problem is to sort the list of attributes according to the accuracy of their classification during the prediction phase, and try replacing the more accurately predicted attributes first whenever applicable. Another concern is that we do not know the number of noisy attributes and/or class in each instance. We try to make as few changes as possible so as to minimize the risk of introducing additional noise into the data.

A third consideration is the criterion of accepting a classification. The 10 classifiers obtained from the 10-fold cross validation of the training data are used to determine the classification. Since these classifiers are built based on noisy data, they might not be a very accurate evaluation mechanism, even when the test instance does not contain noise.[1] Thus we do not necessarily require every single classifier to agree on the classification to accept it as correct. Rather, we determine a reasonable threshold by approximating it with the maximum number of classifiers that agree on the most instances.

In summary, when there are multiple combinations of changes that would fix the classification error, we prefer the one with the fewest number of changes and involving attributes that have been the most accu-

[1] Otherwise, we wouldn't have the need to improve their performance, would we?

rately predicted. In addition, we allow for some slight disagreement between the classifiers in determining whether an instance is considered to have an "acceptably correct" classification. The polished data set such produced can then be used to generate a classifier for future use.

3 EXPERIMENTAL SETUP

In this section we outline the way the experiments were conducted. We used the decision tree builder c4.5 [Quinlan, 1993] as the underlying machinery for classification.

3.1 DATA SETS

Six data sets from the UCI Repository of machine learning databases [Murphy and Aha, 1998] were used. Below we briefly describe some of the characteristics of these data sets. Where separate training and test sets exist at the repository, we combined the two into a single data set. All the attributes are nominal (or ordered), and attributes for identification purposes only were ignored.

mushroom This data set consists of 8124 instances of mushrooms. There are 22 attributes describing the physical characteristics of each instance, and the target concept is the edibility of the mushroom.

soybean The soybean data set consists of 683 instances. There are 35 attributes describing the appearance and climatic conditions. The target concept classifies each instance into having one of 19 diseases.

LED-24 This data set describes the LED display of a digit. The target concept is one of the ten digits $(0 \ldots 9)$, and the attributes are the 7 LED components plus 17 irrelevant attributes with randomly generated values. The data set we generated contains 1000 instances.

vote This data set contains 435 voting records of the House of Representatives in the United States in 1984. The 16 attributes corresponds to issues voted on in the House, and the target concept is the political party affiliation (republican or democrat) of the members.

audiology The audiology data set consists of 226 instances. There are 69 attributes, each corresponding to a property or condition of the case, and the target concept consists of 24 possible audiological diagnostics of the instances.

promoters This data set consists of 106 sequences of 57 nucleotides, and the target concept is whether a gene sequence is a "promoter" (a section of a sequence at which gene expression is initiated).

The characteristics of the data sets are summarized in Table 1. We also give the baseline classification accuracy of c4.5 on these data sets when no noise has been added.

Table 1: Summary of Data Sets

Data Set	Size	#Attributes	#Classes	c4.5 Accuracy
mushroom	8124	22	2	100.0%
soybean	683	35	19	92.1%
LED-24	1000	24	10	100.0%
vote	435	16	2	94.7%
audiology	226	69	24	78.0%
promoters	106	57	2	75.6%

LED-24 is an artificial data set, but it has the advantage that we know exactly the structure of the data set and the relationship between the attribute and class values. This proved to be quite useful for the analytical exercises in Section 5.

3.2 NOISE

We divided each data set into 10 (almost) equal parts. For each run, 9 parts of the data were used for training, and the tree generated from the polished data was used to classify the reserved 1 part of test data. The training set was artificially corrupted by introducing random noise into the attributes as well as the class. A noise level of $x\%$ means that the value of each attribute and the target class is assigned a random value approximately $x\%$ of the time, with each alternative value being approximately equally likely to be selected. The actual percentages of noise in the various components of the data sets are given in Table 2.

The actual percentage of noise is always lower than the theoretical noise level, as sometimes the random assignment would pick the original value. Note that, however, even if we exclude the original value from the random assignment, the extent of the effect of noise is still not uniform across all components. Rather, it is dependent on the number of possible values in the attribute or class. As the noise is evenly distributed among all values, this would have a much smaller effect

Table 2: Noise Characteristics of Data Sets at Various Noise Levels. An instance, attribute value, or class value is considered noisy if the respective component differs from that of the original instance.

Data Set	Noise Level	Instances with Noise	Attribute Noise	Class Noise
mushroom	0%	0.0%	0.0%	0.0%
	10%	91.0%	7.4%	5.1%
	20%	99.2%	15.0%	10.1%
	30%	100.0%	22.4%	15.3%
	40%	100.0%	29.9%	20.7%
soybean	0%	0.0%	0.0%	0.0%
	10%	94.7%	5.6%	8.8%
	20%	99.7%	11.3%	17.3%
	30%	100.0%	16.7%	29.1%
	40%	100.0%	22.5%	38.1%
LED-24	0%	0.0%	0.0%	0.0%
	10%	93.4%	5.2%	8.2%
	20%	99.6%	10.0%	19.2%
	30%	100.0%	14.9%	26.2%
	40%	100.0%	20.3%	35.8%
vote	0%	0.0%	0.0%	0.0%
	10%	81.6%	6.5%	5.3%
	20%	97.7%	13.0%	10.3%
	30%	99.5%	20.0%	16.6%
	40%	100.0%	26.6%	16.3%
audiology	0%	0.0%	0.0%	0.0%
	10%	100.0%	5.2%	8.8%
	20%	100.0%	10.4%	19.9%
	30%	100.0%	15.3%	27.9%
	40%	100.0%	21.1%	38.9%
promoters	0%	0.0%	0.0%	0.0%
	10%	99.1%	7.6%	6.6%
	20%	100.0%	14.3%	10.4%
	30%	100.0%	22.2%	13.2%
	40%	100.0%	29.9%	29.2%

on attributes with a large number of possible values than those that have, say, only two possible values.

3.3 EVALUATION MEASURES

We made use of several metrics for gauging the effects of polishing. First we looked at the classification accuracy and tree size of the decision trees built from the polished data. Then we compared the original noise-free data, the noise-added data, and the polished data to determine the change in the amount of noise attributable to polishing. We also looked at whether the correct value of a noisy attribute or class was indeed the nominated one obtained during the prediction phase of polishing. For convenience, we will use the term "attribute" to refer to both attributes and class when they are treated equally. Below we will discuss the preliminary experimental results obtained.

4 TREE ACCURACY AND STRUCTURE

The classification accuracy of the decision trees built from data sets with various noise levels, with and without polishing, is summarized in Table 3. Differences that are significant at the 0.05 level using a one-tailed paired t-test are marked with an $*$.

Decision trees built from the polished data performed as least as well as the decision trees built from the uncorrected data in all but two categories (soybean at 0%, and audiology at 10% noise levels). At the lower noise levels (0–10%), the differences in classification accuracy were not significant, which could partly be due to the fact that the accuracy was close to 100% for a few data sets in both the polished and unpolished cases. At the intermediate noise levels (20–30%), the improvements using the polished data were significant for most categories. At the highest noise level tested (40%), the results were more mixed. The differences were significant for four of the six data sets.

Grouping by data sets, improvements for soybean, LED-24, audiology and promoters were more significant. The mushroom data set fared well at some noise levels, but vote seemed to be immune to polishing and did not show any significant improvement at almost all levels. The most dramatic improvements were obtained with LED-24 and promoters at the higher noise levels. We will investigate possible reasons for these behaviors in Section 5 when we look at the actual changes made in the data sets.

The average tree size for each category is also shown in Table 3. In all but one category (mushroom at 30% noise level), the polished data gave rise to trees at least as small as the ones built from the unpolished data, sometimes with a fairly substantial reduction in size. Polishing seems to be able to remove some of the irregularities in the data, allowing a more compact tree to be built from the more uniform data.

5 PROXIMITY TO THE ORIGINAL DATA

Next we dug into the data sets and performed a count of the errors removed as well as introduced (separately) by polishing. We devised three different metrics for measuring the proximity of the polished data to the original noise-free data, and the relevant figures are summarized in Table 4. We will introduce the three metrics in Sections 5.1, 5.3, and 5.4 respectively. Some

Table 3: Classification accuracy (with standard deviation) and tree size without and with polishing. An $*$ indicates a significant difference in classification accuracy at the 0.05 level.

Data Set	Noise Level	Accuracy ± Standard Deviation without	with		Tree Size without	with
mushroom	0%	$100.0 \pm 0.0\%$	$100.0 \pm 0.0\%$		30.6	30.6
	10%	$99.9 \pm 0.1\%$	$100.0 \pm 0.0\%$		214.3	173.5
	20%	$99.7 \pm 0.4\%$	$99.9 \pm 0.1\%$	$*$	268.3	228.5
	30%	$99.0 \pm 0.7\%$	$99.3 \pm 0.6\%$	$*$	338.7	414.6
	40%	$98.6 \pm 0.5\%$	$98.7 \pm 0.5\%$		535.2	439.4
soybean	0%	$92.1 \pm 2.0\%$	$91.9 \pm 2.3\%$		95.1	94.5
	10%	$86.2 \pm 4.9\%$	$88.9 \pm 3.0\%$		156.9	123.2
	20%	$83.0 \pm 3.3\%$	$85.2 \pm 4.5\%$	$*$	202.3	192.5
	30%	$72.2 \pm 6.1\%$	$76.3 \pm 4.1\%$	$*$	278.5	243.3
	40%	$50.7 \pm 8.4\%$	$55.0 \pm 4.6\%$	$*$	328.8	299.6
LED-24	0%	$100.0 \pm 0.0\%$	$100.0 \pm 0.0\%$		19.0	19.0
	10%	$100.0 \pm 0.0\%$	$100.0 \pm 0.0\%$		78.2	48.1
	20%	$92.3 \pm 4.3\%$	$97.6 \pm 2.1\%$	$*$	193.4	87.9
	30%	$76.2 \pm 4.7\%$	$90.6 \pm 4.1\%$	$*$	336.0	157.4
	40%	$49.0 \pm 5.9\%$	$67.8 \pm 5.7\%$	$*$	491.6	323.6
vote	0%	$94.7 \pm 2.0\%$	$94.7 \pm 2.5\%$		14.5	4.9
	10%	$94.7 \pm 2.5\%$	$95.0 \pm 3.0\%$		17.8	12.7
	20%	$94.0 \pm 2.1\%$	$95.7 \pm 2.4\%$	$*$	20.8	10.0
	30%	$92.9 \pm 3.2\%$	$94.2 \pm 3.6\%$		43.3	25.6
	40%	$92.4 \pm 3.1\%$	$92.9 \pm 3.0\%$		27.1	18.7
audiology	0%	$78.0 \pm 7.7\%$	$78.4 \pm 8.5\%$		50.5	47.9
	10%	$73.0 \pm 7.9\%$	$72.6 \pm 5.7\%$		66.6	60.6
	20%	$67.8 \pm 8.0\%$	$70.4 \pm 5.8\%$	$*$	90.5	89.1
	30%	$55.3 \pm 11.7\%$	$60.6 \pm 8.3\%$	$*$	127.4	115.0
	40%	$33.6 \pm 11.7\%$	$47.3 \pm 4.4\%$	$*$	143.3	131.6
promoters	0%	$75.6 \pm 13.5\%$	$76.7 \pm 17.3\%$		21.1	11.8
	10%	$73.0 \pm 12.5\%$	$81.6 \pm 12.7\%$	$*$	21.8	15.0
	20%	$65.9 \pm 9.0\%$	$82.1 \pm 9.7\%$	$*$	28.2	12.2
	30%	$55.6 \pm 10.3\%$	$70.9 \pm 9.5\%$	$*$	32.2	16.2
	40%	$60.3 \pm 10.3\%$	$64.8 \pm 11.1\%$	$*$	31.8	30.2

of these metrics are inapplicable when the amount of noise is 0% or when the number of changes made by polishing is 0. These entries are marked with an $*$ in the table.

5.1 REDUCTION IN OVERALL NOISE

An obvious metric to use would be the difference in the amount of noise in the data set before and after polishing. Let n_i and m_i respectively be the number of correct adjustments (from noisy to non-noisy values) and incorrect adjustments (from non-noisy to noisy values) made to the i-th attribute by the polishing procedure. Let t_i be the total number of instances with a noisy value in the i-th attribute. The net percentage reduction in noise averaged over all attributes is given by

$$NR = \frac{1}{k} \sum_{1 \le i \le k} \frac{n_i - m_i}{t_i},$$

where k is the total number of attributes (skipping over attributes with $t_i = 0$). The values for NR are shown in the column *Net Reduction* in Table 4.

The NR score can range from $-\infty$ to $+\infty$, a negative score indicating that more noise has been introduced into than removed from the data set, while a positive score indicates the opposite. Note that in most categories the net percentage reduction, although positive, was not very large. There are a number of excuses for this phenomenon. Recall that even if an instance contained noise, it was tweaked during the adjustment phase only if it could not be classified correctly by the trees originally built using the noisy data in a 10-fold cross validation. We can get an idea of how much noisy data could slip through without being flagged as requiring adjustment, by comparing the classification accuracy without polishing to the actual percentage of instances with noise. Except for the 0% noise level, the percentage of noisy instances was close to 90 or 100%

Table 4: Net percentage reduction in noise, percentage of changes that are correct, and percentage of noise correctly identified during the prediction phase. Ill-defined entries are marked with an $*$.

Data Set	Noise Level	Net Reduction	Correct Change	Noise Predicted
mushroom	0%	*	*	*
	10%	7.4%	69.6%	76.5%
	20%	6.8%	76.6%	73.6%
	30%	13.4%	80.0%	69.7%
	40%	4.2%	69.6%	64.6%
soybean	0%	*	0.0%	*
	10%	7.7%	65.5%	83.2%
	20%	4.0%	88.8%	77.0%
	30%	0.0%	48.6%	69.8%
	40%	0.8%	68.8%	65.6%
LED-24	0%	*	*	*
	10%	9.1%	29.1%	63.6%
	20%	2.1%	28.9%	61.5%
	30%	-2.6%	30.4%	60.0%
	40%	-6.3%	32.1%	57.3%
vote	0%	*	0.0%	*
	10%	1.6%	26.9%	74.1%
	20%	5.3%	47.2%	69.2%
	30%	2.0%	30.6%	65.0%
	40%	4.9%	56.6%	60.2%
audiology	0%	*	0.0%	*
	10%	6.7%	72.2%	92.6%
	20%	4.5%	91.1%	89.7%
	30%	3.1%	83.1%	87.1%
	40%	2.6%	84.6%	85.2%
promoters	0%	*	0.0%	*
	10%	-23.4%	15.8%	26.9%
	20%	-16.9%	14.0%	25.6%
	30%	-7.5%	12.7%	26.2%
	40%	-3.2%	17.1%	27.6%

(Table 2). However, for most categories, the classifier was able to maintain a fairly high classification accuracy without polishing (Table 3). For example, at the 40% noise level, every instance in the mushroom data set contained some noise, but only 1.4% was misclassified. Thus, we can expect a large proportion of the noisy data to go undetected during polishing and therefore was never subjected to adjustment at all.[2]

One of the reasons why the accuracy did not decrease as rapidly as the increase in the amount of noise might be that some of the attributes were not very relevant to the classification. Noise introduced into these attributes would not have as much of an impact. Thus, even though the majority of instances contained at least one noisy value, for most of these instances the classification was not affected, and thus we would not even attempt to adjust their values. Together with our conservative bias towards minimizing the number of adjustments made to each instance, the fairly low percentage reduction in the overall noise is somehow expected in the polishing setting.

[2]Note that the two criteria for correct classification are not exactly the same, as an instance only needs to be classified correctly by one particular tree in cross validation, while it is evaluated against all 10 trees generated from the (10-fold) cross validation in polishing.

5.2 ANOMALY: LED-24

A more disturbing observation is that for LED-24 at the 30–40% noise levels, and for promoters at all noise levels, the amount of noise *increased* after polishing (negative *NR* scores), while their corresponding classification accuracy improvements were among the highest of all data sets. We have two explanations for this phenomenon, and we will verify them using the LED-24 data set. The refined figures for the three metrics of Table 4 are given in Table 5. For comparison, the entries for LED-24 in Table 4 were reproduced in the columns marked *All*.

5.2.1 Irrelevant Attributes

LED-24 contains only 7 "real" attributes, and the other 17 attributes were assigned randomly generated values. We cannot reasonably expect to be able to deal with "noise" in an attribute with random values. By excluding these 17 spurious attributes from the calculation, we obtained the figures shown in the columns *7 Attributes:+Class* in Table 5. The *NR* values showed a marked increase from the ones computed using all 24 attributes. Note that the irrelevant attributes did not have to be removed from the data set before polishing; they were just excluded from the calculation of the metric *NR*.

Thus, we are improving the accuracy of the relevant attributes at the expense of the irrelevant attributes. This arguably can be defended by saying that the relevant attributes are the only ones that really matter. After all, ideally the irrelevant attributes would not enter into consideration by the classifier. However, a procedure that does not put more noise into the irrelevant attributes would still be preferable. One solution might be to limit the adjustments to only those attributes that are actually examined by the classifier. Or we might borrow an idea from the post-pruning of decision trees: after locating an appropriate combination of adjustments for an instance, we can try to remove as many adjustments as possible without al-

tering the classification of the instance. This might limit the spurious changes, as removing these changes should not affect the classification.

5.2.2 Adjustments Towards an Alternative

The other explanation for the low *NR* value of LED-24 has to do with the design of the adjustment procedure. We prefer to change the attribute values rather than the class value, and a change in the latter is considered only when no appropriate combination of attribute changes can be found. Thus, when the class value contains noise, the adjustment procedure would first try to change the attribute values to fit the incorrect class value. The tweaks in the attributes might indeed be the correct ones associated with the altered class value, but in the eyes of our simple counting mechanism, the changes are for the worse as we are introducing additional changes that are not in the original data set. We can expect this effect to be more prominent in data sets where a small number of attributes are highly predictive of the class value, as then only a few tweaks to these attributes would be enough to fit the altered class value.

In the context of LED-24, we can check this hypothesis as we know the structure of this artificial data set. Preliminary examination of the polished data seemed to support the hypothesis. An instance with a changed class value was often prescribed attribute value changes that would move the instance "closer" to the changed class rather than to the original class. Here the distance was measured by the number of differences between the 7 relevant attributes of two instances.

To verify this hypothesis, we applied the metrics of Table 4 only to the portion of the data set whose class values had not been tampered with. These figures are reported in the columns *7 Attributes:-Class* in Table 5. The *NR* values increased to a level not seen in Table 4 for any data sets, but they are more in accord with the extent of improvement in the classification accuracy of LED-24 (Table 3).

We suspect these problems affected all data sets to some extent (perhaps in particular also promoters), but in the other cases it is not as easy to check into the data, as we do not know the underlying structure of the real life data sets.

Thus, the *NR* metric might not be very informative. In particular, we need a better way to assess the amount of noise in a data set. Rather than insisting on the original class value and penalizing any deviation from the original instance, we should reward changes towards fitting the altered class.

5.3 PERCENTAGE OF CORRECT ADJUSTMENTS

Another metric we used was the percentage of adjustments that were correct, in the sense that the corresponding values in the original noise-free data were restored. Let n_i and m_i be as in Section 5.1. The percentage of correct adjustments averaged over all attributes is given by

$$CA = \frac{1}{k} \sum_{1 \le i \le k} \frac{n_i}{n_i + m_i},$$

where k is the total number of attributes (skipping over attributes with zero changes). The values for *CA* are shown in the column *Correct Change* in Table 4.

Unlike the *NR* metric which is expected to have a fairly low value for most data sets, the *CA* metric can have values that range from 0 to 100%. A *CA* value of 50% indicates that the attributes are adjusted correctly half of the time and incorrectly the other half of the time, thus with no net change in the overall amount of noise. We would therefore like to see *CA* values greater than 50%, and the higher the value the better. The data sets mushroom, soybean, and audiology had fairly high *CA* values, indicating that the adjustment procedure was correcting the noisy values quite often. However, LED-24, vote, and promoters had *CA* values well below 50%.

The *CA* metric suffers from the same problems as *NR*. It can be affected by irrelevant attributes, and it does not take into account that the "incorrect" changes might make an instance fit the altered class rather than the original class. The problematic categories seemed to match those we encountered with *NR*. Similar to the situation for *NR*, for the LED-24 data set, there was a substantial improvement in the *CA* values when we considered only the 7 relevant attributes, in particular when we also excluded those instances with noisy class values (Table 5).

5.4 CORRECTLY IDENTIFIED NOISE DURING PREDICTION

One data set that consistently scored low on almost all the metrics we have considered so far is vote. At most noise levels, there was not much significant improvement in classification accuracy, the *NR* values were quite small, and the *CA* values were below 50%. This

Table 5: Statistics for subsets of the LED-24 data set. "All" includes the whole data set. "7 Attributes:+Class" considers only the 7 relevant attributes. "7 Attributes:-Class" in addition excludes all those instances with noisy class values.

Noise Level	Net Reduction			Correct Change			Noise Predicted		
	All	7 Attributes		All	7 Attributes		All	7 Attributes	
		+Class	-Class		+Class	-Class		+Class	-Class
0%	*	*	*	*	*	*	*	*	*
10%	9.1%	26.7%	44.7%	29.1%	72.8%	100.0%	63.6%	97.2%	97.9%
20%	2.1%	18.5%	36.8%	28.9%	70.3%	98.6%	61.5%	91.0%	97.7%
30%	-2.6%	11.0%	26.0%	30.4%	65.2%	94.3%	60.0%	83.0%	89.0%
40%	-6.3%	3.1%	13.6%	32.1%	58.3%	85.6%	57.3%	81.4%	81.9%

prompted us to question the quality of the attribute value nominations available in the first place. In the column *Noise Predicted* in Table 4, we report on the percentage of noisy attribute values in the training set that were correctly identified together with a correct nominated value in the prediction phase.

This value provides an upper bound for how well the adjustment phase can fare, in terms of the metrics we have defined. If a correct attribute value is not on the predicted list, we can hardly make the correct adjustment for it. The percentage of noise correctly predicted was reasonable for almost all data sets, including vote, but not for promoters. We have noted that promoters and LED-24 exhibited similar patterns: high classification accuracy improvement, but negative *NR* scores, and below 50% *CA* scores. Isolating the relevant attributes for LED-24 resulted in a jump in the percentage of noise predicted (Table 5), and we might expect a similar increase for promoters.

The poor results for vote, despite the high percentage of noise predicted, suggests that the weakness was in the adjustment phase of polishing (as opposed to the prediction phase), and in Section 6 we will discuss some desirable extensions. Another possible explanation is that although the percentage of noise predicted was reasonably high, the predictions might not include the most relevant attributes. We need to develop more informative metrics for measuring the various components affecting the performance of polishing, in order to achieve a better understanding of the mechanism.

6 FURTHER THOUGHTS

The analysis of the polished data sets generated during the experimental evaluation and the discrepancies between this and the classification accuracy pointed us to some desirable extensions, both in the polishing procedure and in the evaluation metrics. The prediction phase seems to be adequate, correctly identifying a fair percentage of the noise. There are a number of ways the adjustment phase can be improved. At the moment there is no provision for changing *both* the attribute and class values in the same instance. The mechanism only allows for adjusting either a set of attributes or failing that, the class value alone. Obviously this would miss at least part of all those instances in which both the attribute and class values have been corrupted. We would need to sort out a more comprehensive desirability ordering of the adjustments, and the associated heuristics.

We may also try to "re-polish" the polished data, and see if the data set can be progressively refined. While this might identify additional noise in the data, we can also expect that more and more of the data are *not* adjusted in successive iterations, as the data become more and more uniform. As we have observed, the majority of noisy instances were not adjusted at all, and this proportion is likely to rise with re-polishing. A related problem is whether we can use a more stringent criterion so that more instances would be subjected to adjustment, without overfitting the data to the trees we have at hand. Perhaps trees with a coarser structure (more heavily pruned) would be a better basis for polishing.

Another dimension of improvement is to develop a better evaluation metric for analyzing the performance of polishing. Classification accuracy is a very practical measure, as this is what we would be concerned about with future unseen instances. However, as we have discussed, the measures for determining the actual noise difference in the data might miscount some of the desirable changes as undesirable. Note that in the case of the LED-24 data set, we only tested the hypothesis that the adjustments were made towards fitting the altered class value. In general, however, there might be more than one combination of attribute settings

that can result in the same classification, and the adjustments could well have been made to fit not the original combination, but an alternative but nonetheless correct combination. A more involved metric to determine the error count might be the difference between the polished instance with its nearest neighbor in the original data set, not necessarily the instance it was obtained from.

7 CONCLUSION

The experimental evaluation suggests that polishing is a promising approach to not only identifying the noise in the data, but also correcting it. Trees built from polished data resulted in higher classification accuracy and smaller tree size in many cases. Merely counting the changes in the polished data, however, did not produce very satisfactory scores in a number of cases, and we attribute this partly to the evaluation metrics themselves, which do not fully capture the essential information.

The major drawback of polishing is the time complexity involved. We need to build many trees with each of the attributes as the target class, as well as search over a potentially very long list of predicted changes for the appropriate adjustments. Another weakness is that the procedure is restricted to only nominal attributes, which limits its applicability.

Polishing assumes that the attributes are somehow related to each other and to the target class. We would expect it to perform poorly with two classes of attributes: those that are irrelevant for the classification task, and those that are independent of the other attributes. As discussed in Section 5.2.1, we might minimize changes made to the irrelevant attributes by limiting adjustments to only those attributes deemed predictive by the classifier, or by "post-pruning" the adjustments so that only the essential changes are retained.

In the LED-24 data set, there are 17 irrelevant attributes, and all attributes are conditionally independent given the target class. However, a substantial improvement in classification accuracy could still be obtained, suggesting that the relationship between each (relevant) attribute and the target class might be strong enough in some cases to achieve reasonable classification results. Experimenting with a larger variety of data sets and noise levels and developing more informative evaluation metrics would help us better understand the generality and characteristics of polishing.

Acknowledgement

I would like to thank Uroš Pompe for showing me pictures of Slovenia, and Ross Quinlan for pointing me in the right direction. This research was supported by the Australian Research Council A49801208.

References

[Brodley and Friedl, 1996] Carla E. Brodley and Mark A. Friedl. Identifying and eliminating mislabeled training instances. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996.

[Clark and Niblett, 1989] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

[Domingos and Pazzani, 1996] Pedro Domingos and Michael Pazzani. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 105–112, 1996.

[Gamberger *et al.*, 1996] Dragan Gamberger, Nada Lavrač, and Sašo Džeroski. Noise elimination in inductive concept learning: A case study in medical diagnosis. In *Proceedings of the Seventh International Workshop on Algorithmic Learning Theory*, pages 199–212, 1996.

[John, 1995] George H. John. Robust decision trees: Removing outliers from databases. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 174–179, 1995.

[Kononenko, 1991] Igor Kononenko. Semi-naive Bayesian classifier. In *Proceedings of the Sixth European Working Session on Learning*, pages 206–219, 1991.

[Langley *et al.*, 1992] P. Langley, W. Iba, and K. Thompson. An analysis of Bayesian classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 223–228, 1992.

[Mitchell, 1997] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[Murphy and Aha, 1998] P. M. Murphy and D. W. Aha. UCI repository of machine learning databases. University of California, Irvine, Department of Information and Computer Science, 1998. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[Quinlan, 1987] J. Ross Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3):221–234, 1987.

[Quinlan, 1993] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[Rousseeuw and Leroy, 1987] Peter J. Rousseeuw and Annick M. Leroy. *Robust Regression and Outlier Detection.* John Wiley & Sons, 1987.

[Shapiro, 1987] Alen D. Shapiro. *Structured Induction in Expert Systems.* Addison-Wesley, 1987.