1 Major Research and Education Activities

This project has investigated the effects of systematic data error on inductive classifier learning algorithms. Inductive classifier learning algorithms such as C4.5 and Ripper, among others, have proven to be useful for many applications. Despite their successes, however, these systems are limited by the quality and appropriateness of the data they are given as input.

The process of collecting data is imprecise. Data collections are often riddled with problems, ranging from spurious random noise to systematic error. Over time, classifier learning algorithms have become more robust. It is understood that data will be noisy, and algorithms are developed to handle noise in a variety of ways. However, while work has been done to address the problem of noise, little attention has been given to the treatment of systematic error.

Two general questions have been addressed by this project:

1. How do existing inductive classifier learners handle systematic error in their input data? Is there a difference between the effects of noise and systematic error?

2. Do the results give us direction for developing algorithms to detect systematic error?

In order to address these questions, we developed and conducted an empirical study of the effects of systematic data error on known algorithms. Our work was presented in a talk entitled "Characterizing the Effects of Systematic Data Error on Inductive Classifier Learning Algorithms," given at Purdue University on November 8, 1999 and at the University of Massachusetts, Amherst on December 9, 1999.

1.1 Motivation

Our interest in the issue of systematic error was first motivated by a problem involving the development of a classification system for local telephone network diagnosis. NYNEX's MAX expert system was first put into operation in New York and New England in 1991 (Rabinowitz et al., 1991). MAX (Maintenance Administrator eXpert) serves as a Maintenance Administrator, responsible for analyzing customer-reported telephone problems and dispatching the appropriate type of repair person to fix the problem. The input to the system includes a wide range of attribute values, including the results of electrical tests on the line, the complaint as described by the customer, and the type of switch through which the customer's line runs.

While MAX was working well, it was clear that there was room for improvement, and we were charged with the task of improving the expert system. A difficulty we encountered was that there was evidence (real and anecdotal) of systematic error in the data. For instance, the electrical signature of the customer's line was read and reported by an automated system that required regular calibration. Unfortunately, the system was rarely calibrated. To make matters more complicated, there were multiple such systems in operation at each site, giving us masses of systematically incorrect data that were not necessarily consistent with each other.

Despite the difficulties posed by the domain, the company was motivated to continue work on the problem because of the possible financial benefits of doing so. In the early 90s it was estimated that a single percentage point of improvement in the system would save the company three million dollars a year in incorrect technician dispatches.

Companies now own vast quantities of data that can potentially be analyzed, and those data are likely to contain errors – both random and systematic. Due to the potential benefits of products resulting from these data, it is difficult to dismiss complete domains due to systematic errors in the data.

1.2 Systematic Error

1.2.1 Sources

Systematic error can result from a number of different sources (Beers, 1957), including: (i) errors of calibration of instruments, (ii) experimental (environmental) conditions, (iii) personal errors, and (iv) imperfect technique.

The first of these was a problem we encountered in the MAX domain. The second can be viewed in a number of different ways. It can refer to the fact that measuring devices are often affected by factors such as temperature. It can also refer to the fact that there are certain settings that restrict the type of data collection that can be done (e.g., what types of devices can travel into space or which will operate at high altitudes). It can also refer to the effects of the social or political climate.

The third source of error can arise when an individual has a poor understanding of the problem on which they are working. The fourth more generally describes a situation in which there is a broader lack of understanding of the problem, leading to an approach that is consistent, but incorrect.

1.2.2 Looking at Systematic Error in the Context of Classifier Learning

In the context of classifier learning from data, we can observe systematic errors both in the attribute values and class designations associated with individual instances. While miscalibration of measuring devices will contribute primarily to attribute error, the remaining three error sources can contribute either to attribute or to class error.

1.3 Related Work on the Problem of Noisy Data

While the issue of systematic data error has been largely ignored, there has been a significant amount of work done to address the problem of noisy data.

There have been a number of studies of the effects of noise on classifier learning algorithms. (Aha & Kibler, 1989; Angluin & Laird, 1988; Dietterich, 2000; Okamoto & Yugami, 1996; Quinlan, 1986; Weiss, 1995; among others). Noise may be applied either to the class label or to attributes. $N\%$ random noise signifies that, with probability $N/100$, a value is randomly selected from the remaining alternatives.

Weiss (1995) considers the effects of both random and systematic error in the context of small disjuncts. A small disjunct is a disjunct that correctly classifies few training examples.

Thus “small” refers not to the representation of the disjunct itself, but to its coverage of instances from the training set. Small disjuncts tend to be more error prone than large disjuncts, largely because they tend to be induced from rare or exceptional instances (Holte, Acker, & Porter, 1989). Weiss investigates whether systematic error affects small disjuncts particularly badly. In order to separate the complexities of any given domain from the issues being investigated, he performs his experiments on artificial domains. All experiments are run using C4.5 (Quinlan, 1993). Weiss concludes that attribute noise, missing attributes, and class noise can cause rare cases and small disjuncts to be more error prone than common cases and large disjuncts.

Okamoto and Yugami (1996) give not only an empirical, but a theoretical analysis of the nearest neighbor classifier in noisy domains. Their analysis is an average-case analysis.

More recently, boosting and bagging have been compared for their relative robustness to noise. Dietterich (2000) has found that in the presence of noise, bagging is more robust than boosting.

While some researchers have handled the problem of noise by making their algorithms more robust (for example, Quinlan and Aha), others have focused on identifying and eliminating problematic data (Brodley & Friedl, 1996).

Noisy data have been studied extensively. Their effects have been studied both theoretically and empirically, and learning algorithms have been modified to account for noise. While much remains to be done here, an issue with a greater need for exploration is that of systematic error.

1.4 Setting Up the Study

The goal of our study was to characterize the behavior of current classifier learning algorithms when presented with a training set containing systematic error. In our empirical evaluation, the parameters we varied were:(1) the algorithm, (2) the domain, (3) the type of error in the training set, (4) the amount of error in the training set, and (5) the focus of the error. The last of these refers to the placement of the error within the training set. In a disjunctive concept, for example, was class error introduced across examples covered by all of the disjuncts, or was the error focused on examples covered by a single disjunct?

1.4.1 Algorithms

The algorithms included in our study were C4.5, Ripper (Cohen, 1995), C5.0’s boosting algorithm, a backpropagation neural network algorithm, as well as our own implementation of bagging with ensembles of decision trees. We also performed a limited number of runs with our own implementation of a Naive Bayes learning algorithm.

1.4.2 Domains

We applied the algorithms to both artificial and real domains. We generated four two-class domains and four three-class domains. Of the four two-class domains, one had strictly binary-valued attributes, one had strictly discrete-valued attributes, one had continuous, and one was mixed. The three-class domains were analogous. Generating these artificial

Table 1: Domains used in the study.

Domain	No. Attributes	Types of Attributes	Classes	No. Rules
Art-2 Binary	24	binary	2	6
Art-2 Discrete	24	discrete (5)	2	6
Art-2 Continuous	24	continuous	2	6
Art-2 Mixed	24	1/3 each	2	5
Art-3 Binary	24	binary	3	9
Art-3 Discrete	24	discrete (5)	3	9
Art-3 Continuous	24	continuous	3	7
Art-3 Mixed	24	1/3 each	3	8
Mushroom	22	discrete	2	14
Breast	9	discrete (10)	2	28
RBS	49	discrete (4)	2	393
Faces	460	continuous	4	70

domains allowed us complete knowledge of the domains. We were able to control the initial domain theory as well as the type and degree of error in the training instances.

We also applied the algorithms to a number of real domains, including Breast Cancer, Mushrooms, Ribosome Binding Sites (RBS), and Faces, in which the goal is to identify whether the subject is looking straight, up, to the left, or to the right.

The domains we selected varied in type and complexity. Table 1 summarizes the domains. The parenthesized numbers in the "Type of Attributes" column indicate the number of possible values for each of the discrete attributes. The "No. Rules" column refers to the number of conjunctive rules in the clean domain theory. For the artificial domains, this number represents the domain theory from which the training and test instances were generated. For the real domains, it represents the number of rules learned by applying Ripper to the domain. We present these numbers to give a sense of the complexity of each domain.

1.4.3 Types of Class Error

We introduced both class error and attribute error into the training sets for each domain, though we did so in separate runs.

The class error we studied can be divided into two main types. In one, we systematically mislabeled selected instances of class A as class B. In the other, we mislabeled instances of class A as class B and also mislabeled class B instances as A.

For the unidirectional mislabeling, we considered two different ways of selecting instances for mislabeling. In the first, we mislabeled a given percentage of instances from the training set as a whole. Note that this is a standard model followed for injecting noise into a data set. We did this for purposes of comparison. In the second, we focused the error in instances covered by specific disjuncts of the mislabeled class. This type of focused error is representative of the kinds of errors that might be made systematically in a real data set.

For the bidirectional mislabeling, all error was focused on examples from specifically selected disjuncts.

In all cases, we considered various percentages of error introduced.

1.4.4 Types of Attribute Error

When introducing attribute error, we considered discrete attributes separately from continuous attributes. For discrete attributes, including binary-valued attributes, we introduced error in the form of *shift up by one*. If the possible values for a given attribute were the ten digits 0–9, an instance of 0 would be replaced by 1, 1 by 2, and so on. Instances of 9 would be changed to an unknown. Our rationale for introducing error of this type is our experience that in keying in data, a typical error that occurs is a shift of the hand that would result in the type of error described.

We studied three levels of error in the training set: error in all attributes; error in the first half of the attributes; error in a single specially selected attribute. This, again, is in keeping with our experience of keying errors occuring over multiple contiguous data item entries.

For continuous attributes, we considered a shift of value by $x\%$ of the range of possible values for the attribute, varying x. This models the type of error generated by miscalibrated measuring instruments, for example. In all cases, we considered introduction of error into varying percentages of training examples.

1.5 Hypotheses

Our intuition was that the following behaviors would be observed in response to the introduction of class error into the training sets. For C4.5 and Ripper, we expected that as more error was added, the response would be a stepwise degradation of the systems when tested on clean data. For C4.5, our rationale was that the leaves of the decision trees induced after pruning would have an error tolerance. For instance, say that at a leaf a given class A is predicted, and say that the estimated confidence at the leaf is 0.9, based on 10 training examples fitting the leaf, with one of those classified incorrectly. Up to four additional examples in that set could be misclassified without the leaf changing, giving the tree a certain tolerance for error. Only when the error introduced exceeded a threshold would the leaf change. Our rationale was analogous for Ripper.

For the boosting and bagging algorithms we expected smoother degradation in response to error added. Our rationale was that because of the distribution of examples in the different models and because of the voting scheme, both boosting and bagging would respond similarly to the way they would respond to noise. Furthermore, based on the findings of Dietterich (2000), we expected that bagging would be more robust in the presence of error than boosting.

1.6 References

Aha, D. W. & Kibler, D. (1989). Noise-Tolerant Instance-Based Learning Algorithms. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 794–799). San Francisco: Morgan Kaufmann.

Angluin, D. & Laird, P. (1988). Learning from Noisy Examples. *Machine Learning*, *2*, 343–370.

Beers, Y. (1957). *Introduction to the Theory of Error.* Reading, MA: Addison-Wesley Publishing Company, Inc.

Brodley, C. E. & Friedl, M. A. (1996). Identifying and Eliminating Mislabeled Training Instances. *Proceedings of the Thirteenth National Conference on Artificial Intelligence* (pp. 799–805). Menlo Park: AAAI Press.

Cohen, W. (1995). Fast Effective Rule Induction. *Proceedings of the XII International Conference on Machine Learning* pp. 115–123). San Francisco: Morgan Kaufmann.

Dietterich, T. G. (To appear). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*.

Holte, R. C., Acker, L. E., & Porter, B. W. (1989). Concept learning and the problem of small disjuncts. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 813–818). San Francisco: Morgan Kaufmann.

Okamoto, S. & Yugami, N. (1996). Theoretical Analysis of the Nearest Neighbor Classifier in Noisy Domains. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 355–363). San Francisco: Morgan Kaufmann.

Quinlan, J. R. (1986). The Effect of Noise on Concept Learning. In R. Michalski, J. Carbonell, & T. Mitchell (Ed.), *Machine Learning: An Artificial Intelligence Approach, Vol. II*. Los Altos: Morgan Kaufmann.

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

Rabinowitz, H., Flamholz, J., Wolin, E., & Euchner, J. (1991). NYNEX MAX: A Telephone Trouble Screening Expert. In R. Smith & C. Scott (Ed.), *Innovative Applications of AI 3*. Menlo Park: AAAI Press.

Weiss, G. M. (1995). Learning with rare cases and small disjuncts. *Proceedings of the XII International Conference on Machine Learning* pp. 558–565). San Francisco: Morgan Kaufmann.

2 Major Findings

The findings of our empirical study are as follows:

1. Systematic class error does affect inductive classifier learning algorithms. The type of effect of systematic class error is similar to the effect of noise. However, the magnitude of the effect is greater for systematic error.

2. From our study it appears that attribute error does not significantly affect the performance of inductive classifier learning algorithms.

3. Of the algorithms studied, Ripper currently appears to be the most robust in the presence of systematic error. In fact, it is possible that error is captured specifically in rules of low precedence learned by Ripper. Future work is planned to further investigate whether Ripper is, in fact, as robust as these studies indicate. One would expect that conclusions made about Ripper would apply to other rule-learning algorithms as well.

Results of our study are presented below. Additional data have been collected and are available upon request.

2.1 Class Error

For class error we present results for the three different types of error investigated: (i) unidirectional, not focused (i.e., noise); (ii) unidirectional, focused; and (iii) bidirectional, focused. We present the results for noise first, looking to those results again later in comparisons with the effects of systematic class error.

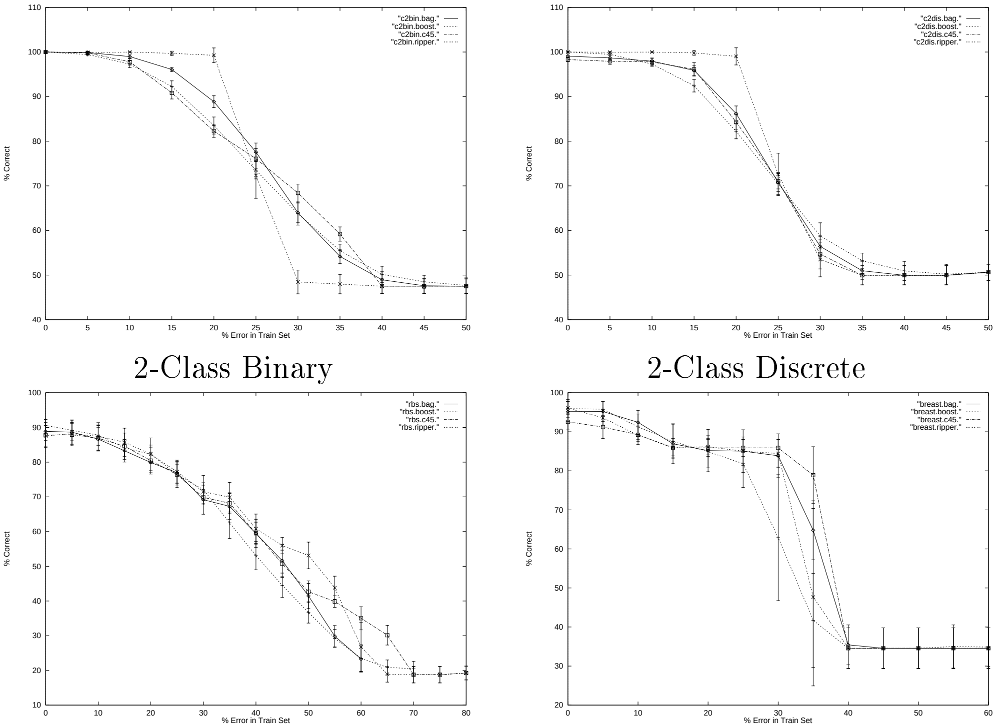
Figure 1 gives results of adding unidirectional, unfocused noise to four domains: Artificial 2-class with binary attributes, Artificial 2-class with discrete attributes, RBS, and Breast Cancer.[1] The x-axis in each case refers to percentage of error introduced into the training set. The y-axis gives the percentage correct on independent test data without error. All results are 10-fold cross-validated. The sizes of the data sets for each of the domains are given in Table 2. All algorithms were run with default settings. For decision tree algorithms, results with pruned trees are given.

The graphs in Figure 1 confirm a number of our hypotheses, but also show other interesting and consistent behaviors. Both the boosting and bagging algorithms, as expected, degrade smoothly as the amount of error in the training set is increased. Furthermore, the percentage error with bagging is lower than that with boosting.

C4.5 degrades smoothly as well, counter to the intuition that the behavior would be stepwise degradation. Upon closer analysis of the trees produced by C4.5, we find that with the addition of error in the data, the trees can change fairly dramatically as C4.5 greedily attempts to fit the data. This behavior has also been documented in another context by Oates and Jensen (1998).

Ripper's behavior stands out, however. As error is added to the training set, Ripper's performance does degrade in a stepwise manner. This is most apparent in the artificial

[1] Only selected results are given here for purposes of readability. The complete set of results is readily available upon request.



2-Class Binary

2-Class Discrete

RBS

Breast Cancer

Figure 1: Behavior of classifier learners in response to class noise.

domains, where its performance on the test set remains consistent up to approximately 20–30% error in the training set. In the case of RBS, the jumps occur more frequently, as would be expected given the relatively low coverage of examples of each of the many rules for that domain.

Not only does Ripper exhibit stepwise behavior in response to error, but it clearly holds out longer in response to error than do the other algorithms. When it does degrade, it performs quite poorly relative to the others. However, in the presence of even up to 20% error, it remains steady.

Figure 2 gives results of adding unidirectional, focused (i.e., systematic) error to the domains presented above. In the case of the artificial two-class binary domain, the graph is for the case where error was a relabeling of the minority class as the majority class, with focus entirely on a single disjunct. In the next two cases, the error was in the direction majority to minority, with focus on two disjuncts. In the fourth graph, we show the results of adding error into two disjuncts separately, and then into both at the same time.

In all cases, the results are analogous to those for class noise. However, the decline in performance occurs earlier – i.e., at lower percentages of error introduced into the training set.

Similarly, Figure 3 gives results of adding bidirectional, focused error to two of the domains. Again, the degradation in performance occurs earlier as the error becomes more focused.

Table 2: Data set sizes.

Domain	No. of examples
Art-2 Binary	5000
Art-2 Discrete	5000
Art-2 Continuous	6437
Art-2 Mixed	4649
Art-3 Binary	5000
Art-3 Discrete	5000
Art-3 Continuous	5735
Art-3 Mixed	7214
Mushroom	8124
Breast	683
RBS	1661
Faces	624

2.2 Attribute Error

In testing the various algorithms on data with attribute error, we found that, with the exception of Naive Bayes, all algorithms were largely unaffected. For up to approximately 30% error, little effect was seen in the performance of the learned classifiers. This was true even in the case where all attributes were affected in a certain percentage of examples from the training set. Figure 4 shows the results of applying Ripper and C4.5 to two domains with attribute error. In both of these cases, all attributes were affected in the percentage of examples indicated.

Figure 5 gives a summary of Ripper's results when applied to the remaining domains in our study. We focus on Ripper due to its strong performance in the presence of class error as well. Note that in the figure, we have removed the error bars for readability.

2.3 Explaining the Results of the Study

The results of applying Ripper to data containing attribute error are particularly surprising in that there is virtually no degradation in predictive accuracy of the learned classifiers in many domains, even when the error is introduced into a large percentage of training examples. This led us to more closely examine the rules output by Ripper. We found that as error is added to the attributes in the training set, Ripper does not stop learning the original "good" rules. Rather, it learns additional rules to cover the instances with perturbed attributes. When these rules are applied to clean test data, the bad rules typically do not apply to the data, even if they have high precedence. This is due to the fact that they refer to attribute value combinations that are not typical in clean data. Thus the original good rules remain those that apply to the clean data.

A similar explanation exists for the behavior of Ripper on data with class error. Again, as noted above, this behavior should be expected from other rule-learning algorithms as well.

Let's call the clean rule set for a domain $RULES0$. When we inject 5% error, $RULES5$

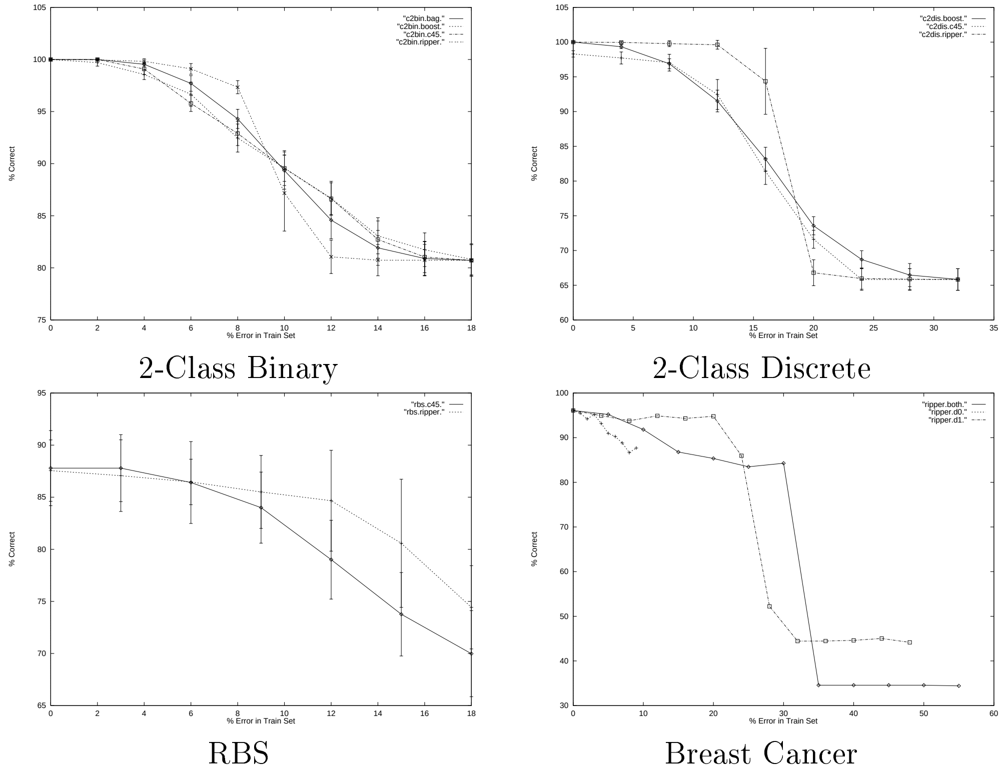


Figure 2: The effect of systematic class error: class A relabeled as class B, with focus of error on specific disjuncts.

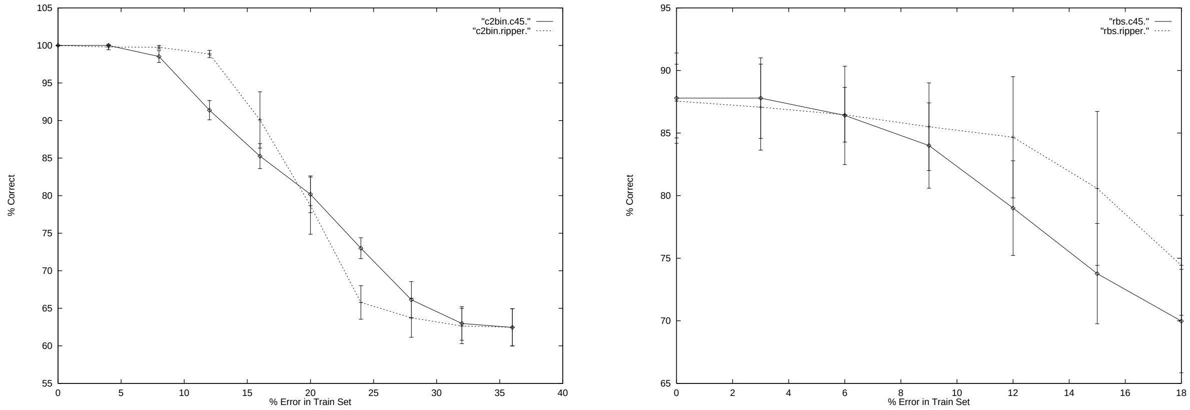
are generated. By comparing $RULES0$ and $RULES5$, it becomes evident that they are the same, with the exception of some new rules added to $RULES5$. Essentially, Ripper generates the same set of rules for $RULES0$ (since there are still enough examples for the algorithm to find the good rules), and then generates some additional rules to cover the errorful examples. At this point, the number of errorful examples is relatively small, so the additional rules fall towards the end of the ordering and therefore do not normally factor into the classification of the test set. The result is a virtually unaffected accuracy.

This is true until the percentage of error injected reaches 15 to 20%. At this point, the rules generated to cover the errorful data have greater coverage than those covering good data. This forces the bad rules higher in the rule ordering, giving the bad rules greater precedence. The net result is a sudden and drastic decrease in accuracy.

2.4 Summary and additional studies

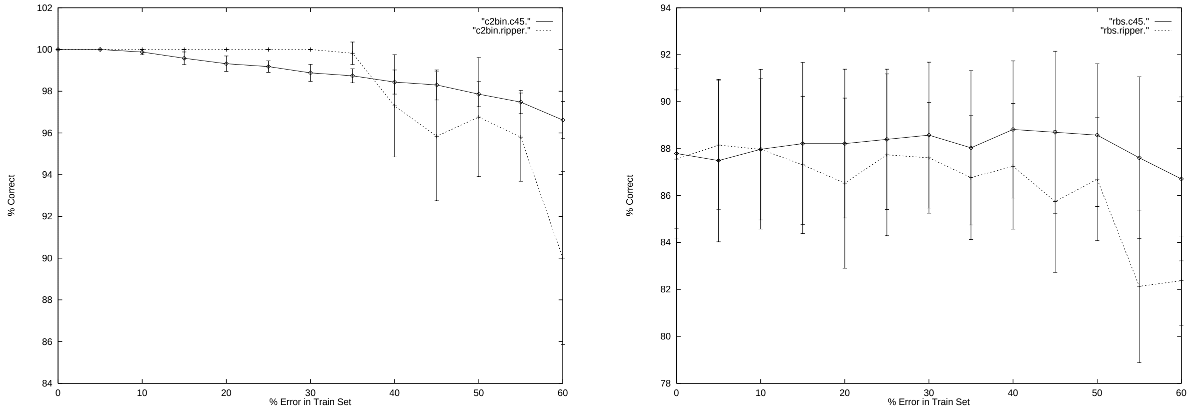
Our conclusions are as follows:

1. Systematic class error does affect the performance of inductive classifier learning algorithms. It appears that the type of effect is analogous to that of noise. However, the magnitude is greater.



2-Class Binary RBS

Figure 3: The effect of systematic class error: class A relabeled as class B and B relabeled as A, with focus of error on specific disjuncts.



2-Class Binary RBS

Figure 4: The effect of systematic attribute error: all attributes affected.

2. Ripper (and thus rule-learning algorithms in general) appear to be most robust in the presence of reasonable (up to 20 or 30%) error. Earlier results had shown similar behavior for RL(Lee, Buchanan, & Aronis, 1998), another rule-learning algorithm.

An interesting observation is that Ripper might be used to analyze a data set and pinpoint any systematic error. Individual rules can be an expression of systematic error in the data. These can be a useful tool for an expert analyzing the data set for possible trouble spots. Before we can strongly make this claim, however, it will be important to investigate this further. This will involve studying Ripper's behavior when given a domain that is inherently filled with small disjuncts, which are likely to be confused with errors. It will also be important to investigate Ripper's performance on more realistic domains than the benchmark datasets available in the Irvine Repository.

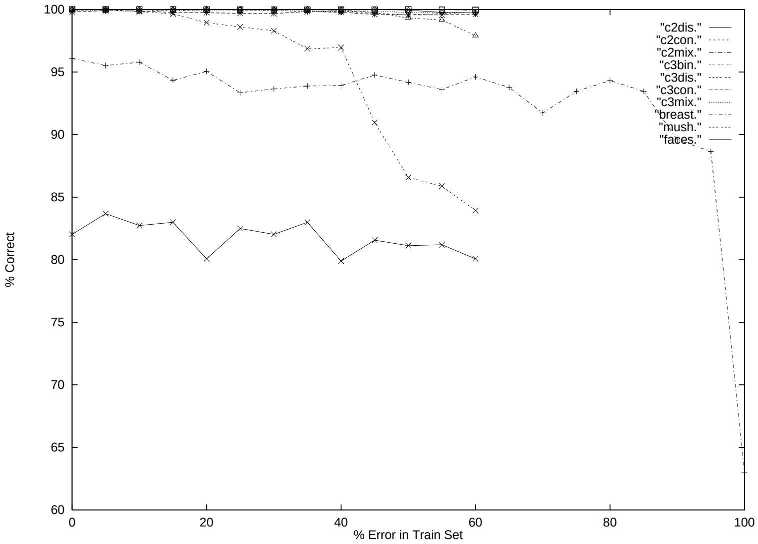


Figure 5: Ripper's response to attribute error: all attributes affected.

2.5 References

Lee, Y., Buchanan, B., & Aronis, J. (1998). Knowledge-Based Learning in Exploratory Science: Learning Rules to Predict Rodent Carcinogenicity. *Machine Learning*, *30*, 217–240.

Oates, T. & Jensen, D. (1998) Large Datasets Lead to Overly Complex Models: An Explanation and a Solution. *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining* (pp. 294–298). Menlo Park: AAAI Press.

3 Opportunities for training and development provided by the project

This project provided research opportunities to three undergraduate students: Alfonso Gonzalez del Riego '00, Esteban Roman '00, and M. Art Munson '01. Alfonso and Esteban worked on the project in the summer before their senior year and Art worked on the project just prior to his junior year. Some early work on the project was also done by Alexander Eaton-Salners '99, who looked into the problem of identifying errorful data during an independent study research course in the fall of his senior year. These students learned a great deal from their involvement in the project.

The project provided these undergraduates with an opportunity to explore research in computer science, an avenue that is not easy to explore through regular coursework. Students in computer science at Williams may elect to take a research course in their senior year. For some, the course continues through the entire year and culminates in a thesis. Summer research, however, provides students with the opportunity to immerse themselves in the work full-time.

This research project involved both theory and empirical evaluation. The students were involved in all phases of the project.

1. The students began by reading background material on inductive machine learning algorithms and the evaluation of those algorithms on noisy data. Of the three summer students, Alfonso Gonzalez del Riego and Art Munson had taken a course in Artificial Intelligence and had some background in machine learning. Esteban Roman, however, had no background in AI before coming to the project.

2. The students met with me regularly during the first week, for lectures and discussions of the work to be performed.

3. The students were involved in the design of the experiments.

4. The students were involved in the selection of data sets.

5. The students implemented software for generating data sets and for gathering results of the experiments. They also implemented some of the learning algorithms studied.

4 Publications

This project and the results described in this report are also available as a technical report from Williams College (see http://www.cs.williams.edu/ãndrea).

5 Products

The software written during the course of this project can be obtained from the PI. The software includes:

1. An artificial domain generator.
2. Error injection software.
3. Software to set up and run experiments of the type discussed in this report.
4. Implementations of bagging over decision trees and Naive Bayes learning algorithms.

6 Contributions

This work has contributed to the machine learning and data mining communities by providing a study of the effect of errorful data on inductive classifier learning algorithms. The algorithms studied by this project are in common use both in research and in applications. Prior to this study, however, no work was done to investigate the performance of these algorithms when applied to data with systematic errors. Of course, as noted above, the effects of noisy data had been considered.

Some have claimed that systematic error is no different from noise. To an extent, this appears to be true as evidenced by our results. However, the effect of focused systematic error is greater than that of equivalent percentages of noise.

Others have claimed that data with systematic error should not be used in learning classifiers, because the integrity of the learned classifier would be in question. This study gives evidence that this depends entirely upon the type of inductive learning algorithm used and the amount of systematic error.

Real, industrial data sets are becoming more readily available all the time. The ability to learn from these data can yield large profits, both intellectually and monetarily. Unfortunately, it is precisely these real data sets that are most probably filled with errors of various types. To dismiss those data sets as uninteresting would be unfortunate, especially in light of the results found in this study.

The software developed in performing this work can be made readily available for further studies. The software includes implementations of learning algorithms that are not generally available, programs for creating artificial data sets, and programs that are the infrastructure for specifying and automatically running the types of experiments described above.