# An investigation of noise-tolerant relational concept learning algorithms

Clifford A. Brunk and Michael J. Pazzani
Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717 USA
brunk@ics.uci.edu

## Abstract

We discuss the types of noise that may occur in relational learning systems and describe two approaches to addressing noise in a relational concept learning algorithm. We then evaluate each approach experimentally.

## 1 INTRODUCTION

Recently, there have been several advances in relational concept learning (Muggleton & Feng, 1990; Quinlan, 1990) that may enable large applications to be implemented (e.g., Dolsak & Muggleton, 1990). A characteristic problem of large learning applications is that there will inevitably be some type of noise in the training data.

In this paper, we first discuss learning when there is noise in the training data. We discuss a variety of types of noise that can occur in relational training data. Next, we describe the FOIL algorithm, concentrating on how it uses an encoding length metric to prevent overfitting the data. We also review reduced error pruning, an approach to dealing with noise that has been successfully applied to decision trees (Quinlan, 1987) and decision lists (Pagallo & Haussler, 1990). Finally, we report on a series of experiments in which we introduce noise into artificially generated training data for a king-rook-king board classification problem. The problem we study is to determine if a chess board containing a white king, white rook, and black king is in an illegal configuration. A configuration is illegal if either king is in check or more than one piece occupies the same space.

## 2 NOISE IN RELATIONAL DOMAINS

Learning from noisy data has been extensively studied in propositional (i.e., attribute-value) domains (Quinlan, 1987; Aha & Kibler, 1989; Mingers 1989). Here, we concentrate on learning from relational data.

In propositional learning systems, a classification error occurs in training data when a datum is not assigned to the correct class. When purposely introducing classification errors into data (for purposes such as testing the noise tolerance ability of a learning algorithm), noisy data is created by assigning some of the training data to a random class. Classification noise maps directly into relational learning systems. For example, in the king-rook-king problem, a 6-tuple ($WK_r$,$WK_f$,$WR_r$,$WR_f$,$BK_r$,$BK_f$) is used to represent the rank and file of the white king, white rook and black king. Therefore, (1,1,1,1,7,5) should be a positive example of the concept illegal (since there are two pieces on the same square). When there is 20% class noise, there would be a 0.20 chance that this example is assigned to a random class (i.e., there is a 0.10 probability it will be falsely reported to the learning program as a negative example).

Attribute-value data may also contain attribute noise, attributes that take on the wrong value. For example, in a medical record a patient's gender might be recorded incorrectly, although the disease of the patient (i.e., the classification) is reported correctly. We will call the generalization of attribute noise to relational learners *tuple noise*. With tuple noise, there is a probability that an element of the k-tuple for a concept to be learned is not correctly recorded. For instance, the previous illegal example might be listed as (1,2,1,1,7,5) rather than (1,1,1,1,7,5). When there is 5% tuple noise, for each element in the k-tuple there would be a 0.05 chance that its value is assigned randomly.

An additional complication of relational learners is that they also require a set of known background predicate definitions as input. These predicates are used as part of the learned concept description. However, there may be noise in these predicates too. Consider the predicate adjacent(X,Y) which is defined extensionally by the set {(1,2)(2,1)(2,3)...(8,7)}. Background classification noise would occur if one of these tuples were identified incorrectly. For example, (1,1) might be included in this set, or (1,2) might be omitted. In addition, background tuple noise would occur if one of the values were recorded improperly. For example, (8,7) might be improperly entered as (8,77). We do not investigate noise in the background predicates further since we assume that the same definitions will be used at learning and testing time.

Eventually, relational learning programs will have to address issues of missing data values as well as noise. A first pass at a taxonomy of missing values can be arrived at by replacing "incorrect" with "unknown" in the previous discussion. However, in this paper we will restrict our attention to just noisy training data.

## 3 THE LEARNING ALGORITHM

FOIL (Quinlan, 1990) is a relational learner which uses a separate and conquer approach guided by an information based heuristic to produce a concept description that covers all the positive examples and excludes all the negative examples. Given a collection of classified examples of a concept (e.g., illegal(A,B,C,D,E,F)) and a set of extensionally defined predicates (e.g., adjacent(X,Y), between(X,Y,Z), equal(X,Y)), FOIL produces a Horn-clause description of the concept in terms of the extensional predicates.

FOIL can be viewed as having two operators: **add-clause** which start a new clause with the body true, and **add-literal** which adds a literal to the end of the current clause body. FOIL performs the **add-literal** operator until no negative examples are covered by the clause, and performs the **add-clause** operator adding new clauses until all positive examples are covered by some clause. FOIL computes the information gain of the variabilizations (i.e., the orderings of existing and new variables) of each extensionally defined predicate in order to determine which literal to add to the end of a clause.

We will report on experiments with FOCL (Pazzani & Kibler; 1990) that extends FOIL by allowing the algorithm to make use of possibly incorrect background knowledge in the form of intentionally defined predicates (i.e., Horn-clause concept descriptions). However, we do not make use of background knowledge in this paper. See Pazzani, Brunk and Silverstein (1991) for one such experiment.

## 4 INFORMATION-BASED STOPPING CRITERIA

Stopping criteria allow a relaxation of the requirement that the concept description cover all positive and exclude all negative examples. This can be useful when learning on noisy data because it provides a systematic method of preventing the algorithm from overfitting the data. An information-based stopping criterion compares some measure of the information required to encode the learned description with some measure of the information required to encode the examples. The result of this comparison determines when to stop learning.

Quinlan (1990) has implemented an information-based approach that attempts to detect when the learner may be overfitting a training set. This formulation determines if the number of bits required to indicate the positive examples the covered by the current clause body minus the number of bits to encode the current clause body is sufficient to allow new literals to be added to the current clause. A similar computation determines if there is enough training data to start a new clause.

EXPLICIT-BITS, the number of bits required to explicitly indicate the positive examples which the clause covers is computed before each literal is added.

$$\text{EXPLICIT-BITS} = \log_2(T) + \log_2\left(\binom{T}{p}\right)$$

T is the number of examples, both positive and negative, in the training set. p is the number of positive examples covered by the clause.

CLAUSE-BITS, the number of bits to encode the clause body is determined after each literal is added to the clause body. CLAUSE-BITS is equal to the sum of the bits to encode each literal in the clause body minus the number of bits to indicate the possible permutations of the literals

$$\text{CLAUSE-BITS} = \left(\sum_{i=1}^{n} 1 + \log_2(R) + \log_2(V_{r_i})\right) - \log_2(n!)$$

n is the number of literals in the clause body. R is the number of predicates. $V_{r_i}$ is the number of possible variabilizations of the predicate used in literal i.

With this stopping criteria, only literals which require less than EXPLICIT-BITS - CLAUSE-BITS bits to encode are considered. Literals are added to the clause until either the clause covers no negative examples or all literals require too many bits. Similarly, clauses are added to the concept description until either no positive example in the training set is left uncovered or all literals require more than EXPLICIT-BITS - CLAUSE-BITS bits to encode.

FOIL's stopping criteria serves two purposes. First, Quinlan has demonstrated that the stopping criteria are useful when no noise is present in the data, but the representational bias and search strategy of the learning system is not sufficient to create an entirely accurate concept description. In the king-rook-king problem, when using the predicates equal(X,Y), adjacent(X,Y), and less_than(X,Y) FOIL learns an approximate definition of illegal(A,B,C,D,E,F) that is <u>nearly</u> 100% accurate. However, if the predicates equal(X,Y), adjacent(X,Y) and between(X,Y,Z) are used (where between(X,Y,Z) is defined as less_than(X,Y) & less_than(Y,Z)), FOIL can learn a definition of illegal that is 100% accurate (Pazzani & Kibler, 1990). We believe this occurs because the 100% accurate definition makes use of not(between(E,A,C)). When less_than(X,Y) is provided instead of between(X,Y,Z), FOIL cannot create not(less_than(E,A) & less_than(A,C)) because FOIL does not create negations of conjunctions of literals. Instead, it appears that FOIL finds ways to approximate this relationship and the stopping criteria allow it to ignore the exceptions to the approximate relationship. The second purpose of the stopping criteria is to avoid overfitting noisy data. However, this use of the stopping criteria has not previously been subject to systematic experimentation.

## 5 REDUCED ERROR PRUNING

Reduced error pruning divides the set of examples available for creating the concept description into two independent sets. One set, the training set, is used to learn a concept description while the other set, the pruning set, is used to increase the accuracy of the learned concept description. In our experiments, we will use FOCL with no stopping criteria to learn a concept. This version continues adding literals to a clause until no negative examples are covered by the clause or until no predicate has positive information gain. Similarly, it builds new clauses until all positive examples are covered. Then a pruning algorithm that uses the pruning set is run on the learned concept description.

This implementation of reduced error pruning is a modification of the pruning algorithm for decision lists described by Pagallo and Haussler (1990). In our implementation, the pruning algorithm uses two operators to increase the accuracy of the concept description: **delete-last-literal** which deletes the last literal from a clause, and **drop-clause** which drops a clause from the concept description. During one pass the pruning algorithm independently applies each operator to each clause in the concept description retaining the modification which leads to the greatest improvement in accuracy. Multiple passes are made over the learned concept description until all operators, if applied, would result in a decrease in accuracy on the pruning set. At this point pruning terminates and the pruned concept description is returned.

One advantage of reduced error pruning is that it is independent of the algorithm used to learn the concept description. It could be applied to the results of inductive systems such as GOLEM (Muggleton & Feng, 1990); explanation-based learning systems (Mitchell, Keller, & Cedar-Kabelli, 1986) or systems that combine explanation-based and empirical learning such as FOCL (Pazzani & Kibler, 1990; Pazzani, Brunk, & Silverstein, 1991), A-EBL (Cohen, 1990) or IOE (Flann & Dietterich, 1989). However, the **delete-last-literal** operator would need to be replaced by a **delete-any-literal** operator. In FOIL, literals are learned in an order that makes the less expensive **delete-last-literal** operator feasible. This ordering is not present in the other systems.

A disadvantage of reduced error pruning is that the training set must be subdivided into two sets which decreases the number of examples available for learning. Since the algorithm continues to prune until there is a decrease in accuracy, it requires a pruning set in which there is at least one example of each disjunctive clause. It may be difficult on small example sets to distinguish a clause that was learned to cover noisy data from a clause that was learned to cover an infrequently occurring disjunction.

## 6 EXPERIMENTAL RESULTS

Experiments were run on the king-rook-king board classification problem. Three algorithms were compared:

- No Stopping - FOCL with no stopping criteria learning from 100% of the training data.
- Stopping - FOCL with FOIL's stopping criteria learning from 100% of the training data.
- REP - FOCL with no stopping criteria learning from 66.7% of the training data and using the remaining 33.3% for reduced error pruning.

Examples were selected from the domain to conform to a 50% positive 50% negative distribution. This was accomplished by randomly selecting the example class, either positive or negative, with probability 0.5, and then drawing examples at random from the domain until obtaining one of the selected class. Noise was then introduced.

The 50% positive 50% negative distribution of examples was chosen to evenly distribute errors of omission and errors of commission in the training set. In the illegal domain as selected from a uniform distribution of chess board configurations, there are many more negative examples than positive. This would skew the classification noise introduced on training data toward falsely classifying negative examples as positive. Since we are interested in a technique which is equally adept at dealing with both types of errors, we normalized the distribution of training and test data.

In the experiments, we ran a number of trials of each algorithm on training data sets of size 80, 160, 320 and 480. Each successively larger set built on the examples of the smaller sets. For instance, the 160 example set contains all the data of the 80 example set and the 320 example set contains all the data of the 160 example set. Each point in a graph represents the mean over all trials as measured by testing on 1000 noise free examples. The bars on the data points represent 95% confidence intervals around the mean. Some confidence intervals have been omitted to avoid clutter.

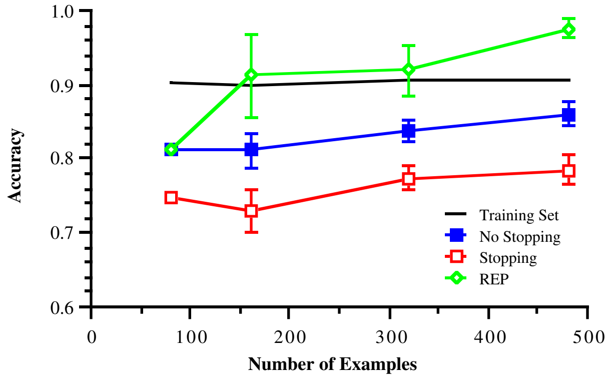


Figure 1. 20% Classification Noise Accuracy

### 6.1 CLASSIFICATION NOISE

In the first experiment, we ran 10 trials of each algorithm. The graph in Figure 1 represents the mean accuracy over 10 trials as measured by testing on 1000 noise free examples. In this run, there was 20% classification noise added to the training data.

The graph shows that reduced error pruning is able to prune a concept definition that overfits 66.7% of the training data, and achieve an error rate smaller than the amount of noise in the domain (since we test accuracy on noise-free data). However, with a small training set (80), this algorithm tends not to significantly improve accuracy. This is due to dividing the data into separate training and pruning sets. With 80 total examples, the pruning set typically contains 13 positive training examples and a clause is deleted unless it is needed to correctly classify one of these examples.

The information-based stopping criteria does not result in the creation of a concept definition that performs as well on this training set. In fact, with the stopping criteria, the system performance is slightly less accurate than overfitting the noisy data by using no stopping criteria. Figure 2 shows the errors of commission and Figure 3 shows the errors of omission for this experiment.

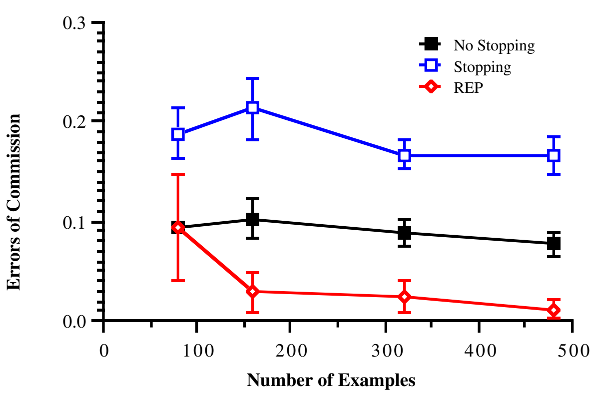


Figure 2. Classification Noise Errors of Commission

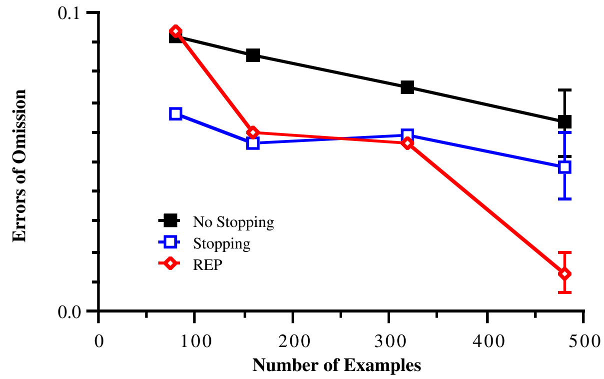


Figure 3. 20% Classification Noise Errors of Omission

These graphs illustrate that with larger data sets, reduced error pruning tends to have fewer errors of omission and errors of commission than overfitting the noisy data. However, the information-based stopping criteria tends to have fewer errors of omission than errors of commission. We suspect that reduced error pruning performs better on this data than the information-based stopping criteria due to its use of two independent samples of the data.

### 6.2 TUPLE NOISE

In the Second experiment, we ran 6 trials of each algorithm on training data sets of size 80, 160, 320 and 480. The graph in Figure 4 represents the mean accuracy over 6 trials as measured by testing on 1000 noise free examples. In this run, there was 5% tuple noise added to the training data. (i.e., For every element of each tuple there is a .05 chance that it has been randomly assigned a value in the range [1..8].) Since each tuple in this domain is composed of 6 elements and there is a .04375 chance that any element is assigned an erroneous value, it follows that there is .2354 chance that a tuple is incorrectly reported to the learning algorithm. Note that a tuple containing an erroneous element value does not necessarily mean that the tuple is misclassified.

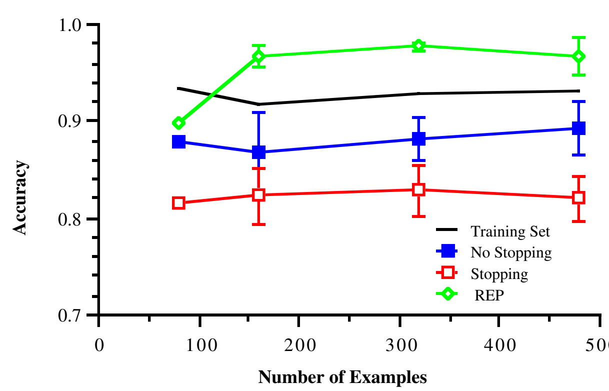


Figure 4. 5% Tuple Noise Accuracy

Figure 5 shows the errors of commission and Figure 6 shows the errors of omission for this experiment. This data show the same pattern as the classification noise experiment.

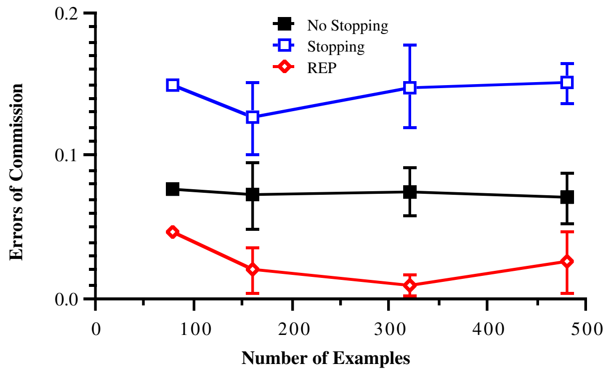


**Figure 5.** 5% Tuple Errors of Commission

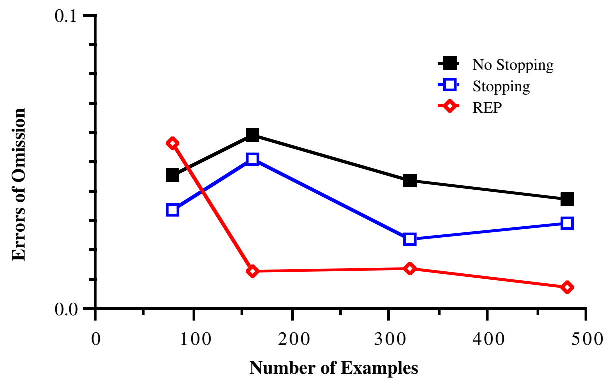


**Figure 6.** 5% Tuple Noise Errors of Omission

## 7 CONCLUSION

We feel that for any large application to be truly successful it will have to be capable of learning accurately in the presence of noise. This topic has been studied extensively in propositional domains and we feel that it should continue to be studied in relational domains. We have presented a brief discussion of the kinds of noise that may occur in relational data and explored two techniques for handling noisy data, one of which appears to significantly increase the accuracy of the learned concept descriptions. These results should be considered preliminary. We have so far tested in only one artificially generated domain. More experiments on other artificial domains will help to gain further understanding of these algorithms. In addition, experiments on naturally occurring data sets will be important in understanding what types of noise exist in these data sets and which noise tolerant algorithms will be useful in practical applications.


### Acknowledgements

We would like to thank Ross Quinlan for his advice on FOIL, and Dennis Kibler for his discussions on noise in learning. This research is supported in part by NSF grant IRI-8908260.


## References


Aha, D., & Kibler, D. (1989). Noise tolerant instance-based learning algorithms. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence,* Detroit, MI: Morgan Kaufmann.

Cohen, W. (1990). *Abductive explanation-based learning: A solution to the multiple explanation-problem* (ML-TR-29). New Brunswick, NJ: Rutgers University.

Dolsak, B., & Muggleton, S. (1991). The application of inductive logic programming to finite element mesh design. *The First International Workshop on Inductive Logic Programming*, Porto, Portugal.

Flann, N., & Dietterich, T. (1989). A study of explanation-based methods for inductive learning. *Machine Learning, 4*, 187–226.

Mitchell, T., Keller, R., & Kedar-Cabelli, S. (1986). Explanation-based learning: A unifying view. *Machine Learning, 1*, 47–80.

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

Muggleton, S., & Feng, C. (1990) Efficient induction of logic programs. *Proceedings of the First Conference on Algorithmic Learning Theory,* Tokyo, Japan. Ohmsha.

Pagallo, G., & Haussler, D. (1990). Boolean Feature Discovery in Empirical Learning. *Machine Learning, 5*, 71–99.

Pazzani, M., Brunk, C., & Silverstein, G. (1991). *An information-based approach to integrating empirical and explanation-based learning* (Technical Report No. 90–38). Irvine,CA: University of California, Department of Information & Computer Science.

Pazzani, M., & Kibler, D. (1990). *The utility of knowledge in inductive learning* (Technical Report No. 90–18). Irvine,CA: University of California, Department of Information & Computer Science.

Quinlan, J.R. (1986a). The effect of noise on concept learning, in R. Michalski, J. Carbonell, & T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.

Quinlan, J.R. (1986b). Induction of decision trees." *Machine Learning, 1*, 81–106.

Quinlan, J.R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies, 27*, 221–234.

Quinlan, J.R. (1990). Learning logical definitions from relations. *Machine Learning,* 5, 239–266.