FOSSIL: A robust relational learner

Johannes Fürnkranz
juffi@ai.univie.ac.at
Austrian Research Institute for Artificial Intelligence
Schottengasse 3
A-1010 Vienna
Austria

Abstract

The research reported in this paper describes FOSSIL, an ILP system that uses a search heuristic based on statistical correlation. Several interesting properties of this heuristic are discussed, and a it is shown how it naturally can be extended with a simple, but powerful stopping criterion that is independent of the number of training examples. Instead, FOSSIL's stopping criterion depends on a search heuristic that estimates the utility of literals on a uniform scale. After a comparison with FOIL and mFOIL in the KRK domain and on the mesh data, we outline some ideas how FOSSIL can be adopted for *top-down pruning* and present some preliminary results.

1 Introduction

Being able to deal with noisy domains is a must for learning algorithms that are meant to learn concepts from real-world data. Significant effort has been made into investigating the effect of noisy data on attribute-value learning algorithms (see e.g. [Quinlan, 1993, Bratko and Kononenko, 1986, Breiman *et al.*, 1984, Mingers, 1989a]). Not surprisingly, noise handling methods have also entered the rapidly growing field of *Inductive Logic Programming* [Lavrač and Džeroski, 1993]. LINUS [Lavrač and Džeroski, 1992] relies directly on the noise handling abilities of decision tree learning algorithms, others, like mFOIL [Džeroski and Bratko, 1992a] and REP [Brunk and Pazzani, 1991], have adapted well-known methods from attribute-value learning for the ILP framework.

This paper presents FOSSIL, a FOIL-like algorithm that uses a search heuristic based on statistical correlation (section 2). One of the nice features of this heuristic is that it gives a reliable measure of the heuristic value of a literal on an absolute and uniform scale. We show empirically that this feature can advantageously be used to deal with noise by cutting off all literals that have a heuristic value below a certain threshold (section 3). We also present empirical evidence that this threshold is robust, in the sense that a good value for it is independent of the number of training examples and of the amount of noise in the data (section 4). After comparing FOSSIL to FOIL and mFOIL we introduce several ideas for adapting pruning methods from decision tree learning in a top-down fashion along with some preliminary results (sections 5 and 6) and finally draw some conclusions (section 7).

2 FOSSIL's search heuristic

2.1 The Correlation Heuristic

FOSSIL's evaluation function is based on the concept of statistical *correlation*. The *correlation coefficient* of two random variables X and Y is defined as

$$corr(X,Y) = \frac{E((X-\mu_X)(Y-\mu_Y))}{\sigma_X \times \sigma_Y} = \frac{E(X \times Y) - \mu_X \times \mu_Y}{\sigma_X \times \sigma_Y} \quad (1)$$

where μ and σ are *expected value* and *standard deviation*, respectively, of the random variables X and Y.

This *correlation coefficient* measures the degree of dependence of two series of points on a scale from -1 (*negative correlation*) to $+1$ (*positive correlation*). In the following description of its adaptation as a search heuristic for the Inductive Logic Programming algorithm FOIL, we will follow the notational conventions used in [Lavrač *et al.*, 1992].

Suppose FOSSIL has learned a partial clause c. Let the set of tuples T_c of size $n(c)$, containing $n^{\oplus}(c)$ positive and $n^{\ominus}(c)$ negative instances, be the current training set. We arbitrarily assign the numeric values $+1$ and -1 for the logical values *true* and *false*. The variable X in (1) now represents the multiset $V(c)$ of the signs (truth values) of the tuples in T_c. The variable Y denotes the multiset $V(L)$ of the truth values of a candidate literal L. A literal L is said to be *true*, whenever there exists a tuple in T_c that satisfies L; if L introduces new variables, they must have at least one instantiation that makes the literal true. Note that $V(c)$ and $V(L)$ naturally contain the same number of values.

The expected values in (1) will be estimated by the mean values of $V(c)$ and $V(L)$ respectively. Standard deviation will be approximated by the empirical variance. Thus we get

$$n = n(L) = n(c) = n^{\oplus}(c) + n^{\ominus}(c),$$

$$\mu_c = \frac{n^{\oplus}(c) - n^{\ominus}(c)}{n}, \ \ \mu_L = \frac{n^{\oplus}(L) - n^{\ominus}(L)}{n},$$

$$\sigma_c^2 = E(V(c)^2) - E(V(c))^2 = 1 - \mu_c^2, \ \ \sigma_L^2 = 1 - \mu_L^2$$

The last remaining term to be computed is $E(V(c) \times V(L))$. If both the truth values $v(c)$ and $v(L)$ of a tuple and the literal under scrutiny have the same sign, then $v(c) \times v(L) = 1$. Conversely, if one is positive and the other negative we have $v(c) \times v(L) = -1$. If we denote the number of positive tuples yielding a negative value for the literal L with $n^{\oplus}(c)^{\ominus}$ (and analogously define $n^{\oplus}(c)^{\oplus}$, $n^{\ominus}(c)^{\oplus}$ and $n^{\ominus}(c)^{\ominus}$), we get

$$E(V(c) \times V(L)) = \frac{n^{\oplus}(c)^{\oplus} + n^{\ominus}(c)^{\ominus} - n^{\ominus}(c)^{\oplus} - n^{\oplus}(c)^{\ominus}}{n}$$

The partial results of above now only need to be substituted into the formula for the correlation coefficient (1). As μ_c and σ_c only need to be evaluated once for each tuple set T_c, evaluation of this formula is not as complicated as it may seem at first sight. Also notice that with this approach no separate calculation for negated literals has to be performed, as a high negative correlation indicates a high dependence on the negated literal.

The literal L_c with the highest absolute value of the correlation coefficient (or $\neg L_c$ if the sign of the coefficient is negative) is then chosen to extend c to form a new clause c'. This is based on the assumption that its high correlation with the current training set T_c indicates some form of causal relationship between the target concept and L_c. The set T_c is then extended to a new set of tuples $T_{c'}$ (which in general will have a different size) and the process continues as described in [Quinlan and Cameron-Jones, 1993].

2.2 Interesting features of the Correlation Heuristic

The information gain heuristic used in C4.5 [Quinlan, 1993] and Foil has been extensively compared to other search heuristics in decision tree generation [Mingers, 1989b, Buntine and Niblett, 1992] and Inductive Logic Programming [Lavrač *et al.*, 1992]. The general consensus seems to be that it is hard to improve on this heuristic in terms of predictive accuracy in learning from noise-free data. While our results confirm this, we nevertheless claim that Fossil's evaluation function has some important features that distinguish it from the weighted information gain heuristic used in Foil.

- In Foil, the heuristic value of each literal and of its negation have to be calculated separately. Fossil does this in one calculation, as positive correlation indicates a causal relationship between the tuple set and the literal under scrutiny, while negative correlation indicates a causal relationship between the tuple set and the negation of the literal.

- The correlation function is symmetric and gives equal consideration to covering many positive and excluding many negative examples.

- The correlation between a tuple set and a literal that has at least one true grounding for each tuple[1] is undefined, because μ_L will be 1 and thus σ_L will be 0. This allows the user to take care of the problem in a flexible way. The experiments reported in this paper ignored this problem by treating undefined cases as having correlation 0. Defining the heuristic value of determinate literals as 1 would put all determinate into the clause body. Irrelevant literals could be removed later in a post-processing phase. Values between 0 and 1 result in the behavior described in [Quinlan and Cameron-Jones, 1993]: until a literal with a correlation above a pre-set value is found, determinate literals will be added to the clause body.

- FOSSIL's correlation coefficient — after taking absolute values and choosing the appropriate, positive or negative, literal — allows to compare the candidate literals on a uniform scale from 0 to 1.

How this last property of the correlation heuristic can be used for a simple, but powerful criterion to distinguish noise from useful information will be described in the next section.

3 The Cutoff Stopping Criterion

The value of FOIL's evaluation function is dependent on the size of the tuple set. The same literal will have different information gain values in different example set sizes of the same concept, although its relative merit compared to its competitors will be about the same. FOSSIL on the other hand can judge the relevance of a literal on an absolute basis. This allows the user to require the literals that are considered for clause construction to have a certain minimum correlation value — the *cutoff*.

This can be used as a simple, but robust criterion for filtering out noise, as it can be expected that tuples originating from noise in the data will only have a low correlation with predicates in the background knowledge. If no literal with a correlation above the *cutoff* can be added to the current clause, this clause is considered to be complete. Similarily, if no literal can be found that can start a new clause, the concept definition is considered to be complete. Note that it may happen that FOSSIL "refuses" to learn anything in cases where no predicate in the background knowledge has a significant correlation with the training data.[2]

[1]This is a super-set of Quinlan's *determinate literals*, but it causes the same problems as described in [Quinlan, 1991].

[2]This has actually happened several times, and is evident in the result with 50% Noise (i.e. random classification) in table 2, where FOSSIL did not learn a single clause in any of the 10 training sets.

If a clause that cannot be further extended still covers negative examples, FOSSIL follows a simple strategy: If the clause covers more positive than negative examples, it is retained, and the examples that are covered will be removed from the tuple set. If the clause covers more negative than positive examples, it will not be added to the concept description, and only the positive examples that would have been covered by this clause will be removed. This is in contrast to FOIL, where learning stops entirely as soon as a clause is found that covers less than 80% positive examples. In that case FOIL leaves the remaining positive examples uncovered, while FOSSIL further thries to find clauses that cover some of them. The fact that the learned clauses always have to cover more positive than negative examples guarantees that the algorithm used in FOSSIL can never produce a bigger error on the training set than the method used in FOIL. It was mainly this improvement that lead to a relatively good performance of FOSSIL at tests on the mesh data (see section 4.5).

4 Experimental Evaluation

4.1 Setup of the Experiments

For the experiments in this paper we have used the domain of recognizing illegal chess positions in the KRK end game [Muggleton *et al.*, 1989]. The goal is to learn the concept of an illegal white-to-move position with only white king, white rook and black king on the board. The goal predicate is `illegal(A,B,C,D,E,F)` where the parameters correspond to the row and file coordinates of the pieces in the above order. Background knowledge consists of the predicates `X < Y`, `X = Y` and `adjacent(X,Y)`[3]. A more elaborate description of this domain and a correct domain theory can be found in the Appendix. Typing constraints were used to speed up the search and recursion was not allowed for efficiency reasons.

Class noise in the training instances was generated according to the *Classification Noise Process* described in [Angluin and Laird, 1988]. In this model a noise level of η means that the sign of each example is reversed with a probability of η. Note that this differs from most of the results in the ILP literature, where a noise level of η means that, with a probability of η, the sign of each example is randomly chosen. Thus a noise level of η in our experiments is roughly equivalent to a noise level of 2η in the results reported in [Lavrač and Džeroski, 1992, Džeroski and Bratko, 1992b]. Noise was added incrementally, i.e. instances which had a reversed sign at a noise level η_1 also had a reversed sign at a noise level $\eta_2 > \eta_1$. Similarly, training sets with n examples were fully contained in training sets with $m > n$ examples.

In all experiments the induced rules were tested against sets of 5000 randomly chosen instances. It also proved useful to record the number of clauses in the

[3]Our definition of `adjacent` actually was `adjacent_or_equal`.

induced concept and the average number of literals per clause to measure the complexity of the learned concept description.

4.2 Finding a good Cutoff Value

The first series of experiments aimed at determining an appropriate value for this parameter for further experimentation. 10 training sets of 100 instances each were used at three different noise levels (5%, 10% and 20%). 6 different settings for the cutoff parameter C were used. The results averaged over the 10 runs are reported in table 1 and plottet in figure 1.

Noise		Cutoff					
		0.0	0.1	0.2	0.25	0.3	0.4
5%	Accuracy	93.05	93.05	93.32	93.58	95.57	93.86
	Clauses	6.3	6.3	6.2	5.8	4.2	2.7
	Lits/Clause	2.25	2.25	2.25	2.19	2.02	1.87
10%	Accuracy	87.77	87.77	90.0	93.44	93.52	83.18
	Clauses	8.2	8.2	6.3	4.5	3.8	1.8
	Lits/Clause	2.74	2.74	2.52	2.24	2.24	1.53
20%	Accuracy	80.21	80.21	85.21	86.87	87.00	72.48
	Clauses	11.4	11.4	6.0	4.1	3.2	0.7
	Lits/Clause	3.09	3.09	2.80	2.76	2.67	0.85

Table 1: Experiments with different settings for the *Cutoff*.

The following observations can be made from these graphs:

- A good setting for C in this domain seems to be somewhere around 0.3 for all three noise levels. Coincidentially, the learned concepts are of about equal complexity at this point.
- The curve for the predictive accuracy is U-shaped, similar to some results from Decision Tree learning (see e.g. [Breiman *et al.*, 1984]).
- There is a transition from overfitting the noise to over-generalizing the rules. A low setting of C has a tendency to fit the noise, because most of the literals will have a correlation above the threshold.[4] Conversely, a too optimistic setting of C results in over-generalization as too few literals have a correlation above the threshold.
- The complexity of the learned concepts monotonically decreases with an increase of the cutoff parameter.

[4]A setting of $C = 0$ results in learning a 100% correct rule for explaining the training set.

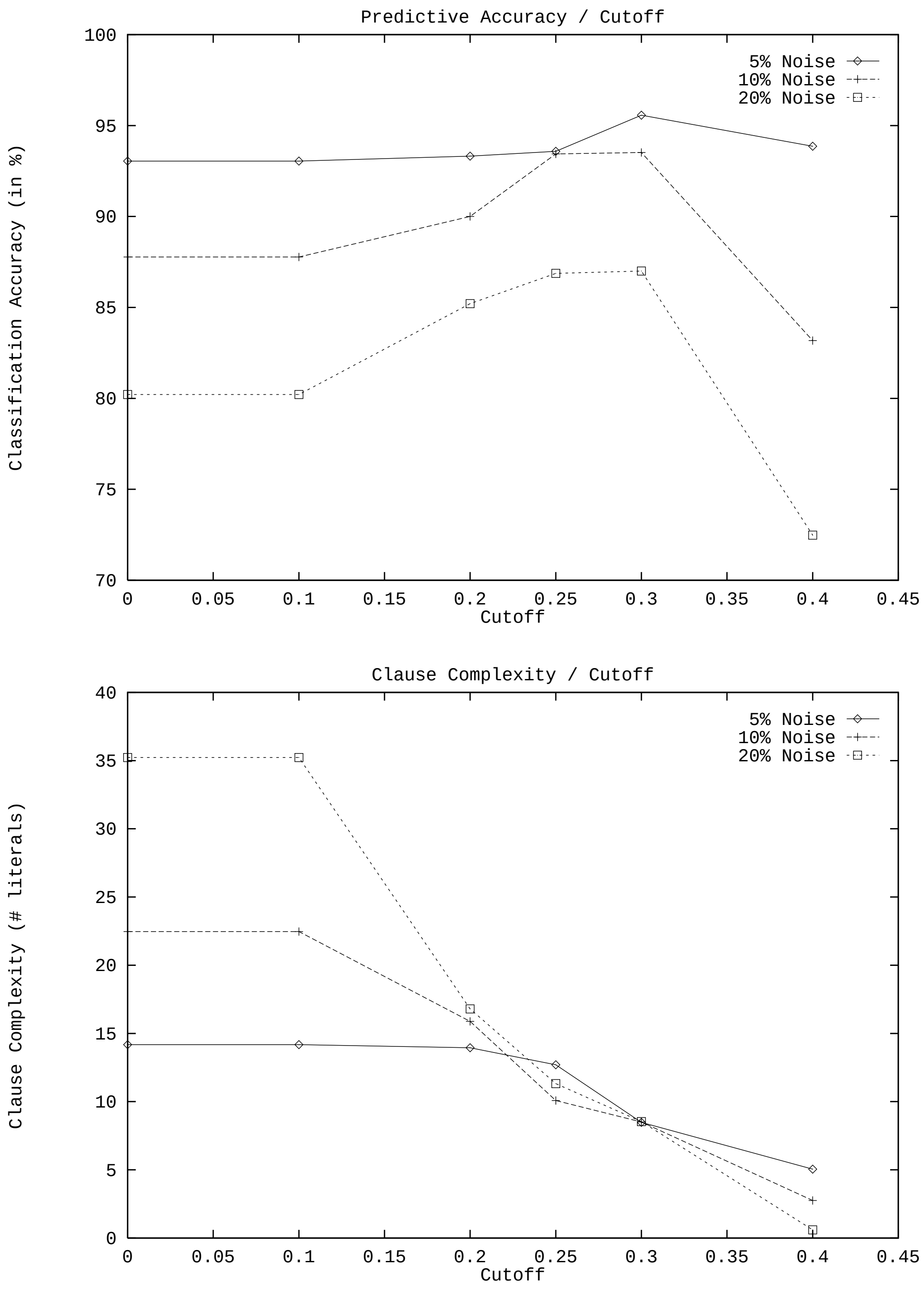


Figure 1: Experiments with different settings for the *Cutoff*.

- The influence of a bad choice of the cutoff is more significant in data containing a larger amount of noise.
- Lowering the setting of the cutoff below 0.1 doesn't seem to change the result.

4.3 Comparison with Foil

We performed two experiments to compare Fossil's performance to the performance of Foil. In the first series we compared the behavior of the two systems with 10 training sets of 100 instances each at different noise levels, which has been the standard procedure for evaluating many ILP systems [Quinlan, 1990, Džeroski and Lavrač, 1991, Džeroski and Bratko, 1992b, Muggleton *et al.*, 1989]. In the second experiment we evaluated both programs at a constant noise level of 10%, but with an increasing number of training instances.

According to the results of the previous experiments we set $C = 0.3$ and never changed this setting.

Comparison at different noise levels

In this experiment we compared Foil4 to Fossil at different noise levels. In order to have a fair comparison to Fossil where backtracking is not implemented, we used two versions of Foil, regular Foil4 and a new version, Foil-NBT, where Foil4's extensive mechanisms of backtracking and regrowing of clauses were not allowed. Surprisingly this version performed better than the original Foil4 in noisy data as can be seen from the results of table 2.

Different Noise Levels		Noise							
		0%	5%	10%	15%	20%	25%	30%	50%
Foil4	Accuracy	98.32	95.26	92.12	90.26	85.21	79.83	71.53	53.00
	# Clauses	3.5	4.2	5.4	5.9	5.7	6.6	8.0	7.9
	Lits/Clause	1.64	1.98	2.41	2.47	2.66	2.98	3.03	3.45
Foil-NBT	Accuracy	98.11	95.00	92.98	91.76	87.12	79.42	76.32	55.33
	# Clauses	3.5	4.1	4.2	4.2	4.5	5.4	5.0	5.2
	Lits/Clause	1.64	1.98	2.34	2.48	2.67	2.80	2.79	3.08
Fossil (0.3)	Accuracy	98.54	95.57	93.52	92.83	87.00	81.63	70.59	(67.07)
	# Clauses	3.7	4.3	3.8	4.2	3.2	2.7	0.7	0.0
	Lits/Clause	1.62	2.02	2.24	2.29	2.67	2.69	0.85	0.0

Table 2: A Comparison of Foil and Fossil on different levels of noise.

An analysis of the result shows that Fossil performs best in most of the tests, but no significant difference between Foil-NBT and Fossil can be found. A comparison of the average number of induced clauses and of the average literals per clause shows evidence that FOSSIL over-generalized at the high noise levels. A lower value of the cutoff parameter may result in better performance in the case of 30% noise, although it is unlikely that a useful theory would be learned. An interesting detail is that FOSSIL did not learn anything at a noise level of 50%, i.e. with totally random data. Thus the cutoff mechanism seems to be a primitive, but efficient means of distinguishing noise from useful information.

On the other hand, FOIL4 seems to perform worse than both FOIL-NBT and FOSSIL. The complexity of the concepts learned by FOIL4 increases with the amount of noise in the data, which is clear evidence for over-fitting noise in the data. The next experiment was designed to confirm this hypothesis.

Comparison at different training set sizes

In this series of experiments we compared FOIL without backtracking to FOSSIL at different training set sizes, each having 10% noise. We decided to use FOIL-NBT instead of FOIL4, because it performed better in the previous series of tests. Besides, the version without backtracking naturally runs faster, which proved to be important. However, we have done a few sample runs with FOIL4 to confirm that its results would not be qualitatively different from those of FOIL-NBT.

Again, we used 10 different training sets and averaged the results. The outcomes of these experiments are summarized in table 3 and figure 2 (the Minimal Error curves will be explained in section 5).

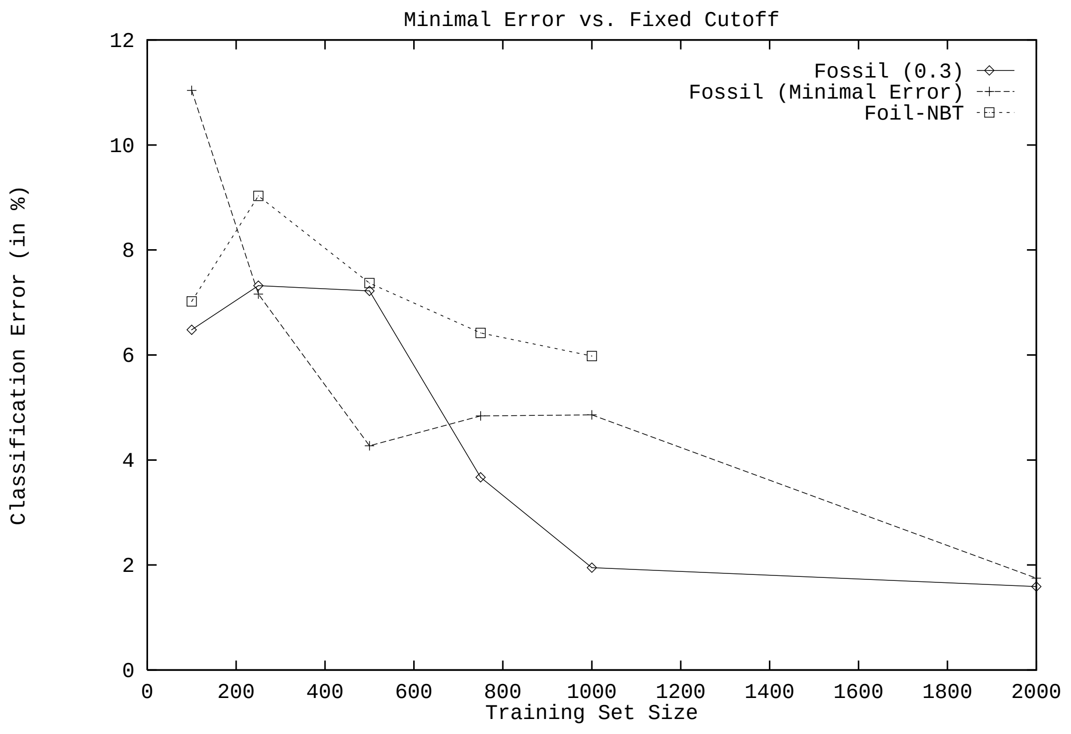
Different Training Set Sizes (10% Noise)		Training Set Size					
		100	250	500	750	1000	2000
FOIL-NBT	Accuracy	92.98	90.97	92.63	93.58	94.02	—
	Clauses	4.2	7.7	11.5	16.7	22.0	—
	Lits/Clause	2.34	3.31	3.61	3.89	4.15	—
FOSSIL (0.3)	Accuracy	93.52	92.68	92.79	96.33	98.05	98.41
	Clauses	3.8	3.7	3.1	3.0	3.0	3.0
	Lits/Clause	2.24	3.01	2.63	1.94	1.5	1.4

Table 3: A Comparison of FOIL and FOSSIL with different training set sizes

The most important finding is that FOIL clearly fits the noise, while FOSSIL avoids this and learns a slightly over-general, but much more useful theory instead. FOIL's fitting the noise has several disadvantages:

Accuracy: The more examples there are in the noisy training set, the more specialized are the various clauses in the concept description, which decreases the predictive ability of each clause learned by FOIL.[5]

[5]This Problem is known as the *Small Disjuncts Problem* [Holte *et al.*, 1989] and has re-



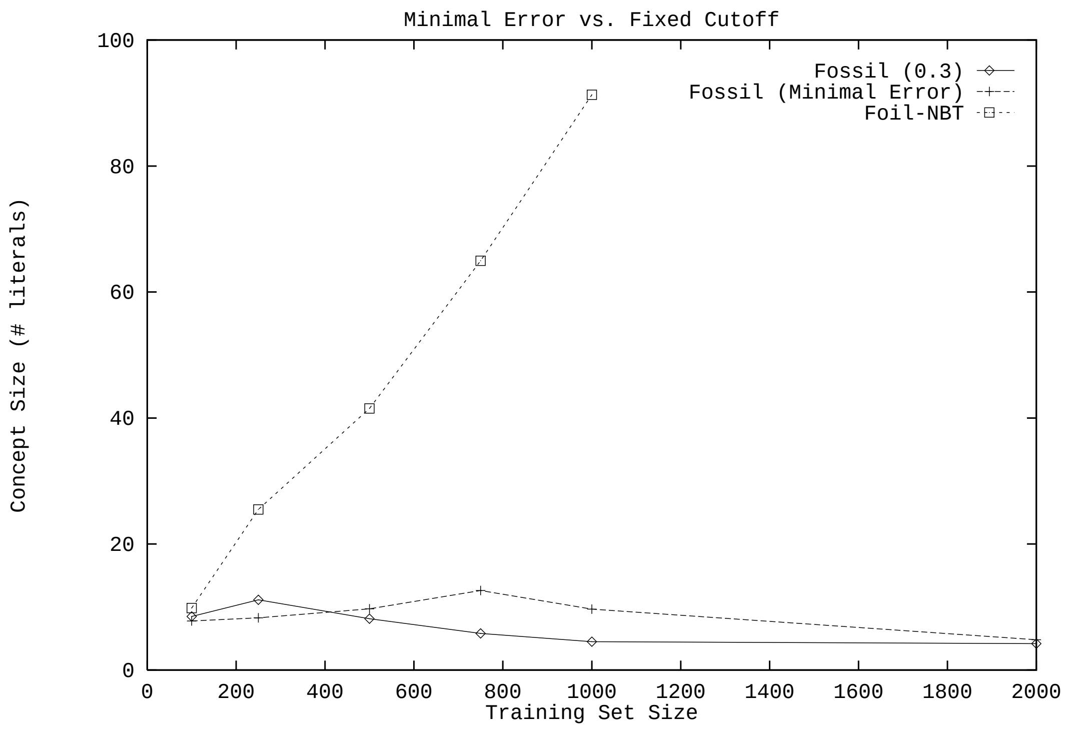


Figure 2: A Comparison of Foil and Fossil with different training set sizes

```
illegal(A,B,C,D,E,F) :- C = E.
illegal(A,B,C,D,E,F) :- D = F.
illegal(A,B,C,D,E,F) :- adjacent(A,E), adjacent(B,F).
```

Figure 3: An approximate theory that is 98.45% correct.

Efficiency: FOIL grows an increasing number of clauses with an increasing number of literals. Also, several of the literals chosen to fit the noise introduce new variables, which leads to an explosion of the size of the tuple set. In fact, the C implementation of FOIL could complete none of the ten experiments with 2000 training examples within 500 minutes of CPU time, while the PROLOG implementation of FOSSIL only needed about 15 minutes of CPU time for each of the training sets, running on the same machine.

Understandability: It is a widely acknowledged principle that the more complex a concept definition is, the less understandable it will be, in particular when both definitions describe the same data set. While the descriptions induced by FOIL for the large training sets were totally incomprehensible to the author, FOSSIL converged towards the simple, approximate theory of figure 3.[6] In fact, in 8 of 10 training sets with 2000 examples exactly this theory was learned, while in the other two the literal `A \== C` had been added to the first clause, which still gives a 97.98% correct theory (see Appendix).

What seems to be responsible for the drastic increase in the complexity of the learned clauses is that FOIL's stopping criterion [Quinlan, 1990] is dependent on the size of the training set. In the KRK domain it performs very well on sample sizes of 100 training examples. The more this number increases, the more bits are allowed for the theory to explain the data. However, more examples do not necessarily originate from a more complex theory. In fact, FOIL very often chooses the same literals as FOSSIL for the first clauses of its concept definition, but then continues to add literals and clauses, where FOSSIL stops.

FOSSIL uses a statistical stopping criterion based on the assumption that each literal in an explanation must have a significant correlation with the set of training examples. Statistical measures usually improve with the size of the training sets and so does the quality of the rules induced by FOSSIL. While both

cently been addressed with an algorithm based on FOIL using probabilistic concept descriptions [Ali and Pazzani, 1993].

[6]This theory correctly classifies all but 4060 of the 262,144 possible domain examples (98.45%). 2940 positions (1.12%)with WK and WR on the same squares and 1120 positions (0.43%) where the WK is between WR and BK on the same row or file are erroneously classified (see Appendix). (Remember that we have defined `adjacent` to mean `adjacent_or_equal`).

FOIL and FOSSIL successively improve their predictive accuracy with increasing training set sizes, only FOSSIL converges towards a useful theory.

4.4 Comparison with *m*FOIL

*m*FOIL [Džeroski and Bratko, 1992a] is an algorithm based on FOIL that has adapted several features from the CN2 learning algorithm, such as the use of the Laplace and m-estimate as a search heuristic and the use of significance testing as a stopping criterion [Clark and Boswell, 1991]. These methods have proved very effective for noise handling. In addition *m*FOIL uses beam search (default beam width 5) and can make use of mode and type information to reduce the search space, features that are scheduled to be incorporated into FOSSIL in the near future. In our experiments *m*FOIL was used to its full capacity.

The values of the m parameter were increased until a maximum performance was reached in the sets of 100 training examples. We then used the same values for testing with 1000 training examples. The results can be found in table 4.

	*m*FOIL					FOSSIL
	0.01	Lap	8	16	32	0.3
100	89.77	89.84	93.03	93.06	91.46	93.52
1000	91.54	92.51	95.70	97.10	98.48	98.05

Table 4: Comparison with *m*FOIL

FOSSIL seems to be at least equal at an example size of 100, unless a considerably better theory has been missed somewhere around $m = 16$. However, *m*FOIL's strengths come to bear at an example size of 1000. The results reported here are probably not yet the peak of its performance, as with $m = 32$ *m*FOIL has learned some theories with a predictive accuracy of above 99% which FOSSIL has not achieved so far.[7] Increasing the m further might well improve the bad theories learned, while keeping the good ones.

However, one of the points to make here is that a good value of the m parameter is not only dependent on the amount of noise (as can be seen from the results given in [Džeroski and Bratko, 1992a] and [Džeroski and Bratko, 1992b]), but also on the size of the example set. Also the values for a good m found in our experiments differ considerably from the ones reported in the above papers (as does the classification accuracy for both FOIL and *m*FOIL). FOSSIL's cutoff parameter on the other hand seems to do reasonably good at different levels of noise and at different training set sizes.

[7]We hope that introducing beam search and using mode and type information will narrow the gap.

In addition, section 5 illustrates some preliminary results for finding good theories without having to specify a good value for the cutoff parameter.

4.5 Experiments with the Mesh Data

We also tested FOSSIL on the mesh data that have been frequently used lately. We followed the same data and the same testing procedure described in [Džeroski and Bratko, 1992a]. The results given there with FOSSIL's results filled can be found in table 5.

	#	FOIL	*m*FOIL	GOLEM	FOSSIL		
					0.10	0.05	0.00
A	55	17	22	17	19	23	24
B	42	5	12	9	11	13	15
C	28	7	9	5	4	6	6
D	57	0	6	11	10	16	10
E	96	5	10	10	2	32	29
Σ	278	34	59	52	46	90	84
%	100	12	21	19	17	32	30

Table 5: Experiments in the Mesh Domain

It should be noted, however, that FOSSIL's good performance in this domain is mostly due to the new minimum precision criterion described at the end of section 3, because *m*FOIL and FOIL both leave a significant amount of positive examples uncovered. The results of FOSSIL were achieved by learning a large amount of very complex rules at low settings of the cutoff.

At higher cutoffs (like 0.3) no rules have been learned, which means that no predicate in the background knowledge was considered to be particularily important for the given classification task. This shows that, although a good value for the cutoff parameter does not seem to depend on the noise level in a domain, it does depend on the explanatory power of the predicates in the background knowledge. That the background knowledge given in the mesh domain is apparently not very good for ILP programs can be seen from the bad results of all programs tried on it. [Džeroski and Bratko, 1992a] discuss some of the problems current algorithms have with this domain and propose some improvements.

The next sections will introduce some ideas how FOSSIL can be used without having to specify a parameter.

```
$C = 1.0$
$Concepts = \emptyset$
while ($C > 0.0$) do
    $NewConcept = Fossil(Examples)$
    $C = MaxPrunedCorr(NewConcept)$
    $Concepts = Concepts \cup NewConcept$
return($Concepts$)
```

Figure 4: Algorithm to generate all Concept Definitions learnable by FOSSIL

5 Generating a series of concept descriptions

5.1 Algorithm

As we have seen in section 4.4, mFOIL and FOSSIL have many similarities. However, a big disadvantage of mFOIL seems to be that it is not so easy to find the right m. The easiest approach is to try the standard settings used in the literature and choose the m that results in the best theory according to an independent test set. However, with this approach one has no guarantee that one does not miss a better theory with a different m. The results given in [Džeroski and Bratko, 1992a] also indicate that the choice of a good m depends on the amount of noise in the data, while our experiments in section 4.4 also suggest a dependence on the size of the training set. FOSSIL, on the other hand, achieved reasonable results with one setting of the cutoff parameter on different noise levels as well as different training set sizes.

Another advantage of the cutoff stopping criterion is — besides its efficiency and stability — its close relation to the search heuristic. While FOIL (*encoding length restriction*) and mFOIL (*significance test*) have to do separate calculations to determine when to stop learning, FOSSIL needs to do a mere comparison between the heuristic value of the best candidate literal and the cutoff value. This allows the design of a very simple algorithm that can generate all theories that could be learned by FOSSIL with any setting of the Cutoff parameter (see figure 4).

The basic idea behind the algorithm given in figure 4 is the following: Assume that you are trying to learn a theory with a Cutoff of 1.0. Unless there is one literal in the background knowledge that perfectly discriminates between positive and negative examples, we will not find a literal with a correlation of 1.0 and thus learn an empty theory. During this run we can remember the literal with the maximum correlation. If we now set the new cutoff to exactly this maximum value, at least one literal (the one that produced this maximum correlation) will be added to the theory.[8] At this new setting of the cutoff parameter we learn a

[8]However, as our experience with FOSSIL shows, this is very often not the only change.

new theory and again remember the maximum correlation of the literals that have been cut off. Obviously, for all values between the old cutoff and this maximum value, the same theory would have been learned and we can choose this value as the cutoff for the next run. It can also be expected that the new theory will be less general than the previous one. This process is repeated until we have the most special theory (with a cutoff of 0.0).

5.2 An Example

In figure 5 we see an example how Fossil generates a series of theories from 1000 noise free examples. It is interesting to see how it steadily improves until it arrives at a 99.32% correct theory. At this point, clauses (1), (5) and (6) try to form a theory of when a position with white rook and black king on the same file is correct. Clause (1) is correct, while clauses (5) and (6) only fit the examples in the training set (for a correct theory see Appendix). Clause (2) on the other hand, says that all positions with white rook and black king on the same rank are illegal, which is too general. After the next step of refinement, Fossil now discovers a new rule (2) which is symmetric to rule (1). In this theory it "forgets" about the already learned clauses (4) to (6) of the last theory, because the starting literals of those rules do not have a high enough correlation under the new circumstances (rule (2) has changed). This goes hand in hand with a decrease in predictive accuracy. Lowering the cutoff once again, however, recovers all of this rules and generates a pretty accurate theory, which completely explains all of the training examples. Consequently no further refinement is possible.

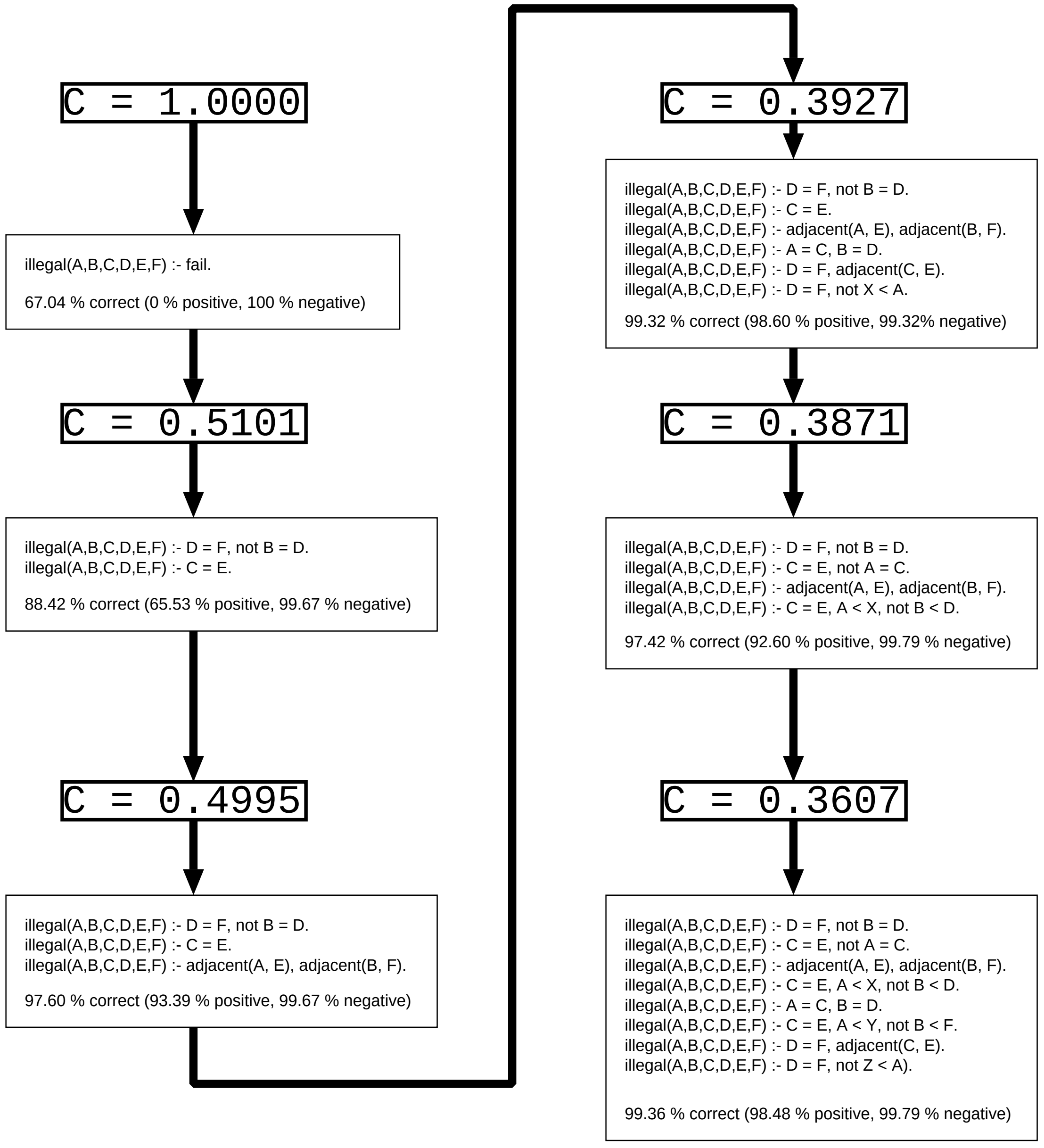
5.3 Experiments

We have used the simple algorithm of figure 4 in the following way: The training sets were randomly split into two sets of equal size, one for training, one for testing. From the training set a series of theories was learned (all theories down to a cutoff of 0.15[9]) and from these the one with the best predictive accuracy on the test set was selected as the final theory. The results — labeled with *Minimal Error* — can be found in figure 2.

It can be seen that this naive and simple method performs better than Foil, although it practically only learns from half of the training examples. However, it is not as good as Fossil with a fixed cutoff. The fact that the minimal error method only uses half of the training examples for learning can also be seen from

Usually several more literals that have a correlation value higher than the new cutoff will be added, because adding a literal to the current concept definition will change the search space for subsequent literals. An example can be found in the next section.

[9]This restriction was only made because of efficiency reasons. From our experience with previous tests we know that theories below 0.15 are usually very specialized and can be expected to give a high classification error.



```
illegal(A,B,C,D,E,F) :- fail.

67.04 % correct (0 % positive, 100 % negative)
```

```
illegal(A,B,C,D,E,F) :- D = F, not B = D.
illegal(A,B,C,D,E,F) :- C = E.

88.42 % correct (65.53 % positive, 99.67 % negative)
```

```
illegal(A,B,C,D,E,F) :- D = F, not B = D.
illegal(A,B,C,D,E,F) :- C = E.
illegal(A,B,C,D,E,F) :- adjacent(A, E), adjacent(B, F).

97.60 % correct (93.39 % positive, 99.67 % negative)
```

```
illegal(A,B,C,D,E,F) :- D = F, not B = D.
illegal(A,B,C,D,E,F) :- C = E.
illegal(A,B,C,D,E,F) :- adjacent(A, E), adjacent(B, F).
illegal(A,B,C,D,E,F) :- A = C, B = D.
illegal(A,B,C,D,E,F) :- D = F, adjacent(C, E).
illegal(A,B,C,D,E,F) :- D = F, not X < A.

99.32 % correct (98.60 % positive, 99.32% negative)
```

```
illegal(A,B,C,D,E,F) :- D = F, not B = D.
illegal(A,B,C,D,E,F) :- C = E, not A = C.
illegal(A,B,C,D,E,F) :- adjacent(A, E), adjacent(B, F).
illegal(A,B,C,D,E,F) :- C = E, A < X, not B < D.

97.42 % correct (92.60 % positive, 99.79 % negative)
```

```
illegal(A,B,C,D,E,F) :- D = F, not B = D.
illegal(A,B,C,D,E,F) :- C = E, not A = C.
illegal(A,B,C,D,E,F) :- adjacent(A, E), adjacent(B, F).
illegal(A,B,C,D,E,F) :- C = E, A < X, not B < D.
illegal(A,B,C,D,E,F) :- A = C, B = D.
illegal(A,B,C,D,E,F) :- C = E, A < Y, not B < F.
illegal(A,B,C,D,E,F) :- D = F, adjacent(C, E).
illegal(A,B,C,D,E,F) :- D = F, not Z < A).

99.36 % correct (98.48 % positive, 99.79 % negative)
```

Figure 5: Series of Theories generated from 1000 noise free examples

the graph, where FOSSIL's rather bad learning results at a size of 500 training examples reappear in the curve for the *Minimal Error* method at a training size of 1000 examples, although there is some improvement in the absolute numbers.

An analysis has also shown that the curves for classification accuracy vs. cutoff are shaped similar to figure 1b, which suggests that some form of hill-climbing can be used to search this series of theories without having to generate all of them (see section 6). However, that a naive search for a local maximum may go wrong can be seen from figure 5

6 Further Research: Top-Down Pruning

While the naive approach of section 5 might be too crude to be applied in this way, we do think that these preliminary results have some potential for refinement. In particular we see some relationship to pruning methods used e.g. in [Brunk and Pazzani, 1991] or [Srinivasan *et al.*, 1992]. The major difference, however, is that we get a series of different concept descriptions in a general to specific order (*top-down*) as opposed to pruning methods the generate a most specific theory first and then successively generalize it (*bottom-up*).

We believe that the top-down approach has several advantages:

- With increasing example set sizes and increasing noise levels, generating a most specific starting theory for pruning becomes more and more expensive (as can be seen from the results of FOIL in section 4.3). Generating a simple general theory is much less expensive. In the experiments described in section 5, typically less than 5 theories have to be generated to find the optimum and in particular the most specific and most expensive theories need not be learned.

- Efficiency can be further increased, as a clever implementation doesn't have to learn an entirely new theory. It can use the part of the last theory up to the point where the cutoff of the literal with the maximum correlation has occured.

- Pruning and learning are interleaved in this algorithm and can influence each other.

- In Decision Tree Learning several methods for selecting the best tree from a series of trees pruned to a different degree have been developed [Mingers, 1989a]. We hope that we can adapt some of these methods for relational learning and in particular make them "incremental", i.e. interleave them with the learning process in a way that generates as few unnecessary and expensive theories as possible.

- A weakness of all these algorithms is that they have to use part of the training set for pruning. Due to the robustness of the cutoff parameter we see a chance that a right value for the cutoff might be determined experimentally on parts of the learning set (e.g. with *cross-validation*) and that this information can be used to infer a good value for the parameter for learning from the entire set.

7 Conclusion

The system described in this paper uses a new search heuristic based on statistical correlation along with a simple stopping criterion. We see the main advantages of this approach in its

Efficiency: There is no separate calculation of a heuristic function for negated literals and the amount of computing involved in calculating the stopping criterion is reduced to a mere comparison.

Robustness: A good value of the cutoff parameter seems to be independent of the amount of noise and the number of training examples. It is nevertheless domain-dependent.

Simplicity: In sections 5 and 6 we have outlined some promising approaches how the simplicity of the cutoff parameter and its close relation to the search heuristic might be used to interleave learning and pruning in a novel way.

However, mFOIL seems to do a little better in terms of classification error provided that one can find the optimal value of the m-parameter. Here we believe that implementing a simple beam search may help to narrow the gap.

Acknowledgements

This research is sponsored by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant number P8756-TEC. Financial support for the Austrian Research Institute for Artificial Intelligence is provided by the Austrian Federal Ministry of Science and Research. I would like to thank J. R. Quinlan and R. M. Cameron-Jones for making FOIL4 `ftp`-able, Sašo Džeroski for providing mFOIL and a lot of patience, and Gerhard Widmer for a PROLOG implementation of the FOIL algorithm to start with and for encouraging this research.

References

[Ali and Pazzani, 1993] Kamal M. Ali and Michael J. Pazzani. HYDRA: A noise-tolerant relational concept learning algorithm. In *Proceedings of the Thirteenth Joint*

International Conference on Artificial Intelligence, pages 1064–1071, Chambèry, France, 1993.

[Angluin and Laird, 1988] D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[Bain, 1991] Michael Bain. Experiments in non-monotonic learning. In *Proceedings of the 8th International Workshop on Machine Learning*, pages 380–384, Evanston, Illinois, 1991.

[Bratko and Kononenko, 1986] Ivan Bratko and Igor Kononenko. Learning diagnostic rules from incomplete and noisy data. In B. Phelps, editor, *Interactions in AI and Statistical Methods*, pages 142–153, London, 1986.

[Breiman *et al.*, 1984] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth & Brooks, Pacific Grove, CA, 1984.

[Brunk and Pazzani, 1991] Clifford A. Brunk and Michael J. Pazzani. An investigation of noise-tolerant relational concept learning algorithms. In *Proceedings of the 8th International Workshop on Machine Learning*, pages 389–393, Evanston, Illinois, 1991.

[Buntine and Niblett, 1992] Wray Buntine and Tim Niblett. A further comparison of splitting rules for decision-tree induction. *Machine Learning*, 8:75–85, 1992.

[Clark and Boswell, 1991] Peter Clark and Robin Boswell. Rule induction with CN2: Some recent improvements. In *Proceedings of the 5th European Working Session of Learning*, pages 151–163, Porto, Portugal, 1991.

[Džeroski and Bratko, 1992a] Sašo Džeroski and Ivan Bratko. Handling noise in Inductive Logic Programming. In *Proceedings of the International Workshop on Inductive Logic Programming*, Tokyo, Japan, 1992.

[Džeroski and Bratko, 1992b] Sašo Džeroski and Ivan Bratko. Using the m-estimate in Inductive Logic Programming. In *Logical Approaches to Machine Learning, Workshop Notes of the 10th European Conference on AI*, Vienna, Austria, 1992.

[Džeroski and Lavrač, 1991] Sašo Džeroski and Nada Lavrač. Learning relations from noisy examples: An empirical comparison of LINUS and FOIL. In *Proceedings of the 8th International Workshop on Machine Learning*, pages 399–402, Evanston, Illinois, 1991.

[Holte *et al.*, 1989] R. Holte, L. Acker, and B. Porter. Concept learning and the problem of small disjuncts. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, MI, 1989.

[Lavrač and Džeroski, 1992] Nada Lavrač and Sašo Džeroski. Inductive learning of relations from noisy examples. In Stephen Muggleton, editor, *Inductive Logic Programming*, pages 495–516. Academic Press Ltd., London, 1992.

[Lavrač and Džeroski, 1993] Nada Lavrač and Sašo Džeroski. *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood, 1993.

[Lavrač *et al.*, 1991] N. Lavrač, S. Džeroski, and M. Grobelnik. Learning nonrecursive definitions of relations with LINUS. In *Proceedings of the European Working Session on Learning*, Porto, Portugal, 1991.

[Lavrač *et al.*, 1992] Nada Lavrač, Bojan Cestnik, and Sašo Džeroski. Search heuristics in empirical Inductive Logic Programming. In *Logical Approaches to Machine Learning, Workshop Notes of the 10th European Conference on AI*, Vienna, Austria, 1992.

[Mingers, 1989a] John Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4:227–243, 1989.

[Mingers, 1989b] John Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3:319–342, 1989.

[Muggleton and Feng, 1990] Stephen H. Muggleton and Cao Feng. Efficient induction of logic programs. In *Proceedings of the 1st Conference on Algorithmic Learning Theory*, pages 1–14, Tokyo, Japan, 1990.

[Muggleton *et al.*, 1989] Stephen Muggleton, Michael Bain, Jean Hayes-Michie, and Donald Michie. An experimental comparison of human and machine learning formalisms. In *Proceedings of the 6th International Workshop on Machine Learning*, pages 113–118, 1989.

[Quinlan and Cameron-Jones, 1993] John Ross Quinlan and R. M. Cameron-Jones. FOIL: A midterm report. In *Proceedings of the European Conference on Machine Learning*, pages 3–20, Vienna, Austria, 1993.

[Quinlan, 1990] John Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[Quinlan, 1991] John Ross Quinlan. Determinate literals in inductive logic programming. In *Proceedings of the 8th International Workshop on Machine Learning*, pages 442–446, 1991.

[Quinlan, 1993] John Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[Srinivasan *et al.*, 1992] A. Srinivasan, S. H. Muggleton, and M. E. Bain. Distinguishing noise from exceptions in non-monotonic learning. In *Proceedings of the International Workshop on Inductive Logic Programming*, Tokyo, Japan, 1992.

Appendix: The KRK domain

Introduction

The KRK domain was first used in [Muggleton *et al.*, 1989]. Since then it has become a standard test bed for Inductive Logic Programming algorithms such as FOIL [Quinlan, 1990], GOLEM [Muggleton and Feng, 1990], LINUS [Lavrač *et al.*, 1991], mFOIL [Džeroski and Bratko, 1992a], NM-GOLEM [Bain, 1991], and FOSSIL. The goal is to distinguish legal from illegal positions in a chess endgame with white king, white rook and black king on the board, i.e. to find a correct definition of the predicate `illegal(A,B,C,D,E,F)`, the six arguments of which are the file and row resp. coordinates of the three chess pieces in the above order.

```
illegal( A, B, C, D, E, F) :- A == C, B == D.
    %  (1) WK and WR on same square.
illegal( A, B, C, D, E, F) :- C == E, D == F.
    %  (2) BK and WR on same square.
illegal( A, B, C, D, E, F) :- adjacent( A, E), adjacent( B, F).
    %  (3) WK and BK on adjacent (or equal) squares.
illegal( A, B, C, D, E, F) :- C == E, A \== C.
    %  (4) WR gives check and WK on different row
illegal( A, B, C, D, E, F) :- D == F, B \== D.
    %  (5) WR gives check and WK on different file
illegal( A, B, C, D, E, F) :- C == E, not between( B, D, F).
    %  (6) WR gives check and WK on same row
illegal( A, B, C, D, E, F) :- D == F, not between( A, C, E).
    %  (7) WR gives check and WK on same file

adjacent( X, Y) :- X is Y + 1.
adjacent( X, Y) :- X is Y.
adjacent( X, Y) :- X is Y - 1.

between( Z, X, Y) :- X < Z, Z < Y.
between( Z, X, Y) :- Y < Z, Z < X.
```

Figure 6: A correct theory

A Correct Domain Theory

A correct domain theory in PROLOG clauses is given in Figure 6.

Of course this is only one possibility of defining a theory of this domain. A close examination of the rules e.g. reveals that rule (2) is redundant, as the cases where the black king and the white rook are on the same square are already covered by rules (1), (4) and (5): Either all three pieces are on the same square

(1) or only the black king and the white rook are on the same square and the white king is on a different row (4) or on a different file (5). Nevertheless we stick with this representation, because we feel it is more intuitive (and maybe easier to analyze).

In some domain definitions in the literature, `adjacent` has been defined without its second clause, thus complicating the definition of `illegal`. Also, very often the definition of the `between` relation is not given. We will use the above domain theory for a numerical analysis of the domain, because the basic structure remains the same in other formulations of the theory.

However, in the examples used throughout this paper, the relation `between` was not available in the background knowledge. Instead the relation `<` was provided. In that case rules (6) and (7) will unfold to several other rules, which will only cover a couple of examples and thus are very hard to detect.

Number of positions: As each of the 6 variables can have 8 different values there are $8^6 = 262,144(100\%)$ different instances in this domain.

3 Pieces on same square: Naturally, there are 64 (0.024%) possibilities for having all three pieces on the same square, because there are 64 squares on a chess board.

2 Pieces on same square: There are 64 squares for the first piece, 1 possible square for the second piece and 63 different squares for the third piece. So we have altogether $64 \times 1 \times 63 = 4032(1.538\%)$ possibilities.

Rule (1): From the above results follows that rule (1) of the domain theory covers $64 + 4032 = 4096(1.563\%)$ positions.

Rule (2): As the positions with all three pieces on one square are already covered by rule (1) there are only 4032(1.538%) positions left for rule (2).

Adjacent Kings: Each of the 36 squares on rows 2 to 7 and files `b` to `g` has 8 adjacent squares. This adds up to $36 \times 8 = 288$. The 4 corner squares have 3 adjacent squares each, another $4 \times 3 = 12$ possibilities. The other 24 border squares have 5 adjacent squares each, i.e. $24 \times 5 = 120$. Altogether there are $288 + 12 + 120 = 420$ possibilities for placing two adjacent kings on a chess board.

Rule (3): For each of the adjacent king positions there are 62 open squares for placing the white rook. Including the cases where the kings are on identical squares, which are also covered by this rule, we have rule (3) covers $420 \times 62 + 4032 = 30,072(11.472\%)$ positions.

Rules (4) and (5): For each position of the black king there are 7 possibilities to place the white rook on the same file/row. Depending on whether the black king is in the corner, on the border file/row, border row/file or in the center, we have 54, 53, 52 or 50 possible squares for the white king, not counting cases that have already been covered by rule (3). Thus each of the rules (4) and (5) covers $7 \times (54 \times 4 + 53 \times 12 + 52 \times 12 + 50 \times 36) = 22,932(8.748\%)$ positions.

White King not between Black King and Rook: We assume that all three pieces are on the same file/row, that the two kings are not adjacent and that all pieces are on different squares. If the black king is immediately adjacent to the white rook, there are 5 squares remaining for the white king under the above assumption, except for the case where the black king is on the border, where we have 6 possibilities. So we get $6 \times 5 + 1 \times 6 = 6 \times 6 = 36$ possible positions. The same number results when we swap black king and white rook. If we now increase the distance between black king and white rook by 1, we analogously get $2 \times (5 \times 5) = 50$ more positions. This is repeated until the distance between black king and white rook is 5 squares, when only 1 possible square is left for the white king. So we have a total of $2 \times (6^2 + 5^2 + 4^2 + 3^2 + 2^2 + 1^2) = 2 \times 91 = 182$ positions.

Rules (6) and (7): As the above calculations are the same for each file or row, each of the two rules covers $8 \times 182 = 1,456(= 0.555\%)$ positions.

Summing up: A summary of the results is given in table 6:

Rule	Covered	Percentage
(1)	4,096	1.563%
(2)	4,032	1.538%
(3)	30,072	11.472%
(4)	22,932	8.748%
(5)	22,932	8.748%
(6)	1,456	0.555%
(7)	1,456	0.555%
illegal	86,976	33.179%
legal	175,168	66.821%
total	262,144	100.000%

Table 6: Rule Coverage

Removing the redundant rule (2) would add most of its coverage to either rule (4) or rule (5).

Approximate Theories

Sometimes ILP programs do not learn the complete theory, but instead learn an approximation. Figure 7 gives an approximate theory which has been reported in a similar form in [Quinlan, 1990], [Bain, 1991], [Srinivasan *et al.*, 1992] and in this report (figure 3).

```
illegal( A, B, C, D, E, F) :- A == C, B == D.
    %  (a) same as (1).
illegal( A, B, C, D, E, F) :- adjacent( A, E), adjacent( B, F).
    %  (b) same as (3)
illegal( A, B, C, D, E, F) :- C == E.
    %  (c) generalization of (2), (4), (6).
illegal( A, B, C, D, E, F) :- D == F.
    %  (d) generalization of (2), (5), (7).
```

Figure 7: Approximate Theory A

Theories like this are in particular learned in the presence of noise or because there are not enough training examples.

Theory A: The approximate theory of figure 7 is an over-generalization of the theory in figure 6, in a way that all positions with white rook and black king on the same row/file are treated as illegal. It is wrong for all cases, where the white king is between the white rook and the black king, thus blocking the check.

We already have covered the case where the white pieces are on the same square and the cases where the two kings are adjacent will be handled by rule (b). Assume that all three pieces are on the same file/row. The maximum distance between white rook and black king is when both are on opposite sides of the file/row. The white king then has 5 possible squares for blocking the check, i.e. rules (c) or (d) resp. will consider 5 illegal positions as legal. We get another 5 cases when swapping white rook and black king. If the white rook and the black king are moved towards each other, the number of squares for the white king decreases, but there are more possibilities for putting the black king and the white rook on the board. So e.g. there are 2 possibilities to put the black king and the white rook on one file/row with 5 squares inbetween. Each of them yields 4 valid squares for the white king. Thus we get $2 \times (1 \times 5 + 2 \times 4 + 3 \times 3 + 4 \times 2 + 5 \times 1) = 70$ possibilities for each file/row. Considering that there are 8 rows and 8 files, rules (c) and (d) together erroneously classify a total of $70 \times 8 \times 2 = 560 \times 2 = 1,120 (= 0.427\%)$ of all examples.

Theory B: In the presence of 10% noise FOSSIL converges towards theory A except rule (1) is usually not found (see section 4). This is not surprising, because as we can see from table 6 an example exclusively covered by rule (1) only occurs in 1.56% of the training data. The regularity in these examples can easily be over-looked, when 10% of the examples are erroneously classified. Besides, many of the examples for rule (1) are covered by the three remaining rules, so that the error for leaving out this rule is not that big, as we will see in the following.

Of the 4096 positions with white king and white rook on the same square, $\frac{1}{8}$th ($= 512$) have the black king on the same row. Another 512 have the black king on the same file. Subtracting the 64 positions where all three pieces are on the same square, which we have counted twice now, we get 960 positions that are covered by rules (c) and (d). In addition rule (b) covers positions where the white rook and king are one square diagonal of the black king. This are $36 \times 4 + 24 \times 2 + 4 \times 1 = 196$ positions. So rules (b), (c) and (d) of theory A cover $196 + 960 = 1156$ positions that would otherwise be covered by rule (a). This means that in addition to the error of Theory A, dropping rule (a) misclassifies another $4,096 - 1,156 = 2,940$ positions. The total error of Theory B thus is $2940 + 1120 = 4060 (= 98.451\%)$.

Theory C: Another good approximation results when rules (c) and (d) are replaced with rules (4) and (5). This means instead of generalizing rules (2) (4) and (6) to rule (c) the most common of the three rules is chosen as a representative. The cases that are not correctly classified by this approximation are those, where the white king is on the same row as his rook and the enemy king, but is not inbetween them, the black king thus being in check. We have already seen that rule (2) is redundant, so the only cases that will be misclassified by this approximation are those that would originally have been covered by rules (6) and (7), i.e. $1,456 \times 2 = 2,912 (= 1.111\%)$.

Theory D: Theories B and C can also be interleaved, e.g. only rule (c) is replaced by rule (4). In this case in addition to the 1,456 errors made by missing out rule (6), we have the 560 mistakes by overgeneralizing rules (2), (5) and (7) to rule (c). This means we have a total error $560 + 1,456 = 2,016 (= 0.769\%)$.

Theory E: Of course, dropping rule (a) from Theory C yields another approximation. This amounts to dropping rules (1), (2), (6) and (7) from the correct theory of figure 6. We have already seen that there are $420 + 64 = 484$ possibilities for placing the two kings on adjacent or identical squares. These cases are still covered by rule (b), but the remaining $4096 - 484 = 3612$ positions will be erroneously considered as legal by this theory. In addition the 2,912 mistakes made by Theory D, this yields a total error of $2,912 + 3,612 = 6,524 (= 2.489\%)$ for Theory E.

Theory	Rules	Error	Accuracy
A	(a), (b), (c), (d)	0.427%	99.573%
D	(a), (b), (c), (5) or (a), (b), (4), (d)	0.769%	99.231%
C	(a), (b), (4), (5)	1.111%	98.889%
B	(b), (c), (d)	1.549%	98.451%
F	(b), (c), (5) or (b), (4), (d)	2.019%	97.981%
E	(b), (4), (5)	2.489%	97.511%

Table 7: Accuracy of Approximate Theories

Theory F: The last approximation we consider is dropping literal (a) from the two possible Theories D. Let us assume we have the theory consisting of rules (b), (c) and (5). Of the 4096 positions with white king and white rook on the same square, $\frac{1}{8}$th (= 512) have the black king on the same file. These instances are now covered by rule (5). In addition rule (b) covers all positions with the black king one square above or below this file. Considering different numbers of neighboring squares we get $36 \times 6 + 12 \times 4 + 12 \times 3 + 4 \times 2 = 308$ covered positions. So $4096 - 512 - 308 = 3276$ positions with white king and rook on the same square are not covered by other rules. Also replacing rule (d) with (5) misclassifies 2,016 examples as we have seen above. So we get a total error of $2,016 + 3,276 = 5,292 (= 2.019\%)$.

Summary: A summary of the approximation errors for approximate theories ordered according to their approximation accuracy can be found in table 7.