# A New MDL Measure for Robust Rule Induction

Bernhard Pfahringer

Austrian Research Institute for Artificial Intelligence
Schottengasse 3
A-1010 Vienna
Austria

**E-mail:** bernhard@ai.univie.ac.at

**Area:** Inductive Learning

**Keywords:** Minimum Description Length, Pruning, Noise.

**Abstract**

We present a generalization of a particular Minimum Description Length (MDL) measure that sofar has been used for pruning decision trees only. The generalized measure is applicable to (propositional) rule sets directly. Furthermore the new measure also does not suffer from problems reported for various MDL measures in the ML literature. The new measure is information-theoretically plausible and yet still simple and therefore efficiently computable. It is incorporated in a propositional FOIL-like learner called KNOPF. We report on favorable results in various purely symbolic propositional domains. Both rule quality in terms of simplicity (and syntactic closeness to the respective underlying theory where known) and predictive accuracy of induced theories are convincing.

# 1 Introduction

The *Minimum Description Length (MDL) Principle* [Rissanen 78], sometimes also called the *Minimum Message Length (MML) Principle*, has been successfully applied in Machine Learning, both for inducing decision trees [Quinlan & Rivest 89, Quinlan 93, Wallace & Patrick 93, Forsyth 93], for constructing new attributes [Pfahringer 94], and in ILP [Muggleton et al. 92]. But recently also some problems with MDL were discovered [Quinlan 94].

In this paper we try to explain the origin of these problems in section 2. A new MDL measure applicable to propositional rule learning aimed at overcoming these problems will be introduced in section 3. Section 4 describes a propositional FOIL-like learning algorithm using this new MDL measure as both a stopping criterion for rule induction and as a criterion to choose between different rule sets, especially to choose between sets of pruned rules and sets of unpruned rules. Empirical results are reported in section 5. Section 6 discusses open problems and further research directions.

# 2 MDL in Rule Learning and its Problems

Empirical induction is always faced with the problem of *overfitting* the data, especially in the presence of noise or irrelevant attributes. The MDL principle is a possible solution as it measures both the simplicity and the accuracy of a particalur rule set in a common currency, namely in terms of the number of bits needed for encoding. A very good introduction to MDL and also its close relation to Bayesian theory can be found in [Cheeseman 90]. He defines the message length of a rule set (called *model* in his article) as:

```
Total message length = Message length to describe the model +
                       Message length to describe the data,
                       given the model.
```

This way a more complex theory will need more bits to be encoded, but might save bits when encoding more data correctly. The theory with the *minimal* total message length is also the *most probable* theory explaining the data [Rissanen 78]. Now the problem for machine learning consists of finding the appropriate coding schemes for the particular kinds of models induced, be it decision trees, propositional rules, Prolog programs, or neural networks.

The precise MDL formula used by [Quinlan 93] for simplifying rule sets is:

$$Cost = TheoryCost + log_2\left(\left(\begin{array}{c} C \\ FP \end{array}\right)\right) + log_2\left(\left(\begin{array}{c} NC \\ FN \end{array}\right)\right)$$

In this formula $TheoryCost$ is an estimate for the number of bits needed to encode the theory. $C$ is the total number of training examples covered by the theory, $FP$ is the number of false-positive examples, $NC$ is the total number of training examples not covered by the theory, and $FN$ is the number of false-negative examples. So the second and the third terms of the formula estimate the number of bits needed to encode all false-positive and all false-negative examples respectively. In summary this formula approximates the total cost in the number of bits for encoding a theory and its exceptions.

So what are the problems with that formula? [Quinlan 94] states two and we would like to add an additional one:

1. The formula for computing exception cost is symmetric, i.e. a theory producing consistently wrong classifications (i.e. labeling every positive example as *negative* and vice versa) will have an exception cost of zero, just like a complete and correct theory.

2. If the class to be learned is significantly in the majority (minority), induced theories tend to under-generalize (over-generalize), especially in the presence of noise and with small numbers of learning examples.

3. When learning from lots of examples in the presence of noise, there is still a tendency to fit the noise.

[Quinlan 94] tries to remedy complaints 1 and 2 by introducing an ad-hoc factor penalizing theories that predict distributions too far off the distribution found in the training data. He also briefly investigates two alternatives for encoding exceptions. All these three modifications empirically improve categorical prediction for small data sets in the presence of noise. But none of these can solve problem 3. To motivate the solution proposed below, let us have one more look at the complaints:

1. We don't see this as a problem! On the contrary, we view this as a very nice property, because a theory consistently producing the wrong classification is - at least for the 2-class learning tasks considered here - a very good theory for the *negation* of the concept to be learned, and as such it is a valuable inductive result. Besides, the theories being compared by MDL are typically constructed by algorithms exhibiting a strong bias towards coverage of and accuracy for positive examples. So in practice such *negated* theories will probably not be induced at all (unfortunately, as one might say).

2. A closer look at the exception cost formula reveals the source for over/under-generalization. Assuming that the positive examples form a significant majority, any induced reasonable theory will also have to cover

the majority of the training data (being mostly positive examples). This causes the exception cost to be dominated by its first summand:

$$log_2\left(\left(\begin{array}{c} C \\ FP \end{array}\right)\right)$$

That of course results in false-positive errors being much more expensive than false-negative errors. So when minimizing the total cost, theories producing few errors of commision will be favored. Empirically such theories are overly specific and produce considerably more errors of omission (which are inexpensive, because the total number of examples not covered by the theory is small).

The same kind of reasoning can be applied for the minority case, too.

3. Overfitting when given too many examples is also explained easily. Assume we have a training set of given size and a level of class noise strictly larger than zero.[1] Furthermore assume we know the correct underlying theory, so that we can compute its cost. What happens if we double the number of training examples? The cost of encoding the theory will remain the same, as it is not a function of the number of examples. But exception cost will rise as there will also be 2 times as many errors in total.[2] Thus incorrect, more complex theories (incurring higher theory cost) that partially overfit the data (incurring smaller exception cost) can evaluate to a total bitcost smaller than that of the correct theory.

All these problems with the above formula stem from the fact that this formula is just an approximation of the generic MDL principle as defined above. It does not estimate encoding cost of *all* examples with respect to a given theory but instead computes a kind of penalty for wrong classifications only! Furthermore this penalty function is in general asymmetric in the way it assigns cost for false-positives and false-negatives (It is symmetric for rare theories covering exactly 50% of all training examples). This explains both problems 2 and 3. So the remedy would be to look for a formula that is more faithful to the MDL principle.

[Muggleton et al. 92] describe a (rather complicated) scheme for encoding theories and data in an ILP setting in a way such that both the theory and a single proof for every example are encoded. Their argument is that such a scheme minimizes the sizes of proofs, thus leading to the induction of efficient theories. But

[1] We are using the so-called *Classification Noise Process*[Angluin & Laird 87] where a class noise level of $\eta$ means that for every training example the classification is *reversed* with probability $\eta$. This corresponds to a class noise level of $2\eta$ for the model incorporated by [Quinlan 94]. Note that neither model preserves the original distribution of positive and negative examples in the undisturbed data (unless there are exactly 50% positive examples).

[2] We can estimate: $log_2\left(\left(\begin{array}{c} 2n \\ 2k \end{array}\right)\right) \sim 2log_2\left(\left(\begin{array}{c} n \\ k \end{array}\right)\right)$

we think the essential property of their formula is just to encode every example in terms of the theory. In [Srinivasan et al. 92] they report impressive results (which are reproduced below in table 3 of section 5) for large enough training sets in the KRK domain, a standard ILP testbed. But one can notice shortcomings: their scheme is rather sensitive to the number of training examples and the level of noise. About half of the entries of table 3 are empty, meaning no compression was achieved. We therefore suspect that their coding scheme for theories and proofs is inefficient in an information-theoretic sense: too many bits are needed for encoding the theory and proofs, which in turn causes theories to produce no compression for small training sets.

# 3 An alternative MDL formula

[Forsyth 93] introduces a well-performing formula for encoding decision trees:

$$cost(tree) = \Sigma cost(leaf_i)$$
$$cost(leaf_i) = d_i + e_i * n_i$$

where $d_i$ is the depth of the leaf in the tree, $n_i$ is the number of examples covered by the leaf, and $e_i$ is average entropy of the outcome at that leaf defined by:

$$e_i = -(p * log(p) + (1-p) * log(1-p))$$

where $p$ is the proportion of positive examples covered by $leaf_i$.[3] Note that $e_i * n_i$ is the number of bits needed by an optimal or 'Huffman' coding of the classifications at $leaf_i$ in terms of the relative frequencies of positive and negative examples at $leaf_i$.

We have modified this formula for coding sets of propositional rules. The essential differences are a cost estimate for examples not covered by the rule set and an information-theoretically plausible encoding cost for the rules themselves. Note that the ordering of rules is significant in this encoding, meaning that an example is covered by the first of all the rules matching it. We define the cost of a rule set as follows:

$$cost(ruleset) = n_{nc} * e_{nc} + \Sigma cost(rule_i)$$
$$cost(rule_i) = rc_i + e_i * n_i$$

where $n_{nc}$ is the total number of examples not covered by the rule set and $e_{nc}$ is the according entropy of this set. One can interpret this as the cost for an empty rule added at the end of the rule set classifying all left-over examples as *negative*. So this is the penalty for one kind of error: omissions or false-negatives.

[3] $0 * log(0)$ is defined to equal 0.

The complexity of a single rule is accounted for by $rc_i$ which could be defined in analogy to trees as just the length of a rule. Experiments and some thought showed a serious flaw: in the presence of noise overly complex but correct rules covering only a few examples are still cheap, because their total cost is equivalent to their length. A better solution is to really try to encode the body of the rule. Assuming a total number $N_{pt}$ of tests that could possibly be used by a rule and adopting Quinlan's idea for encoding exceptions we can define the cost for encoding the body of a rule as follows. We just have to choose the appropriate number of tests out of all possibly used tests (remember that order of tests is not important in propositional settings). The cost for choosing $Length_i$ tests out of $N_{pt}$ possible tests now can be estimated as:

$$rc_i = log_2 \left( \left( \begin{array}{c} N_{pt} \\ Length_i \end{array} \right) \right)$$

This rule cost estimate lead to much better empirical results for noisy training sets as reported in section 5. It may also account for the better results reported below when applying our system to some of the domains used by [Forsyth 93] (besides the principle advantages of learning rules instead of trees).

The new estimate certainly solves problem 3 as coding cost for *all* examples is estimated (remember that the entropy of a rule $e_i$ is multiplied by the total number of examples covered by that rule $n_i$ as part of the cost of a rule). Problem 2 is only partially solved as errors are penalized in a totally different way. We do not get consistently over- or under-generalizing behavior, but with too small training sets the empty theory can result from induction. But this is a consequence of using MDL itself: enough positive data has to support a rule, otherwise the intrinsic cost of the rule will outweigh the classification advantage gained by this rule. Regarding the so-called problem 1, the new formula is even more symmetric in the sense that in principle positive and negative rules could be freely mixed in an induced theory. For practical reasons one would have to add one more bit per rule for encoding the decision part (positive or negative) of each rule, if one wanted to take advantage of that property.

To summarize, the new formula measures cost for encoding all the training examples in terms of the theory (the single rules), classification errors are accounted for at a per-rule basis using local entropies, and complexity of rules is estimated in an information-theoretically plausible way. Furthermore this formula still is symmetric with respect to *negative* theories.

# 4 Algorithmic Usage of the new Formula

For empirical testing of the new formula we have implemented a kind of propositional FOIL [Quinlan & Cameron-Jones 93] called KNOPF. Right now KNOPF is restricted to purely symbolic 2-class learning problems. It is completely free

of user-settable parameters. The MDL principle is used in two ways: firstly as a stopping criterion when inducing a single rule set and secondly for choosing the final rule set out of a number of induced rule sets (here implicitly MDL is also used to judge which kind of pruning is necessary - see below). Different rule sets are induced by combining various standard search heuristics for single rule induction with two different forms of pruning as discussed below.

Single rules are constructed top-down by using standard search heuristics [Lavrac et al. 92]: info or accuracy gain, weighted or not, approximating probabilities using either relative frequencies, the Laplace estimate or the m-estimate (for different values of m, currently $0.5, 1, 2, 4$). Construction of a single rule stops when either only positive examples are covered, or the heuristic value cannot be increased any further, or no possible test is left to be incorporated. Such rules are then immediately pruned in one of two ways. One way of pruning is intended for presumably noise-free data, the second way of pruning is intended for presumably noisy training data.

The first pruning strategy is *correctness preserving*: the pruned rule will not cover more negative examples than the unpruned rule. The second strategy just maximizes the difference $p - n$ of positive and negative examples covered by the rule. This strategy for single rule induction is inspired by the IREP-algorithm [Fuernkranz & Widmer 94]. One difference is that we always use all of the available data both for constructing and pruning single rules - no splitting into separate training and pruning sets ever takes place.

A novelty of KNOPF with respect to pruning is the incorporation of a complete search algorithm - namely *branch-and-bound* - to determine the global optimum for both cases of pruning. So single rule induction could also be described as a two-tiered process: an efficient heuristic search using purity measures and all possible tests to determine a small subset of tests is followed by a complete search using only that small subset of tests and one of two very simple evaluation functions.

Induced rules are added incrementally to the current rule set as long as the total cost of the rule set improves, i.e. decreases. Once a rule is induced that does not improve global cost, rule induction is stopped. That way KNOPF performs greedy hill-climbing search for inducing each rule set using the MDL principle as a stopping criterion.

The MDL principle is also used for choosing the final resulting theory. As we straightforwardly induce all theories using the various search heuristics mentioned above combined with both pruning strategies we get a lot of competing theories. The least expensive theory according to the MDL measure is returned as the final result of induction. As that set will be the result of one of the two pruning mechanism, in a sense the system also automatically decides which kind of pruning is necessary, thus implicitly judging the presence of noise in the learning data. When a set of correctness preservingly pruned rules is chosen as the final result, we can assume noise-free data and vice versa.

# 5 Empirical Results

In the following experiments, KNOPF's performance was usually averaged over ten runs choosing training sets at random. The class attribute was randomly reversed (from *yes* to *no* or vice versa) for $N\%$ of the training examples, when the noise level was set to $N$. We always report the accuracy of both KNOPF and C4.5RULES [Quinlan 93] and sometimes quote additional results from the literature.

## 5.1 Various Boolean Concepts

The following three data sets have been used by [Forsyth 93] for comparing his TREEMIN program with various other ways of pruning. We reproduce his description of the sets here:

- RAND. This is simply a random data set containing 12 random binary variables plus a target variable which is 1 in approximately 50% of the cases and 0 in the other 50%. Training set size was 255, test set size 100.

  We do not report any accuracies achieved; that it would not make much sense. The interesting qualitative result is that KNOPF did return the empty theory in 8 of the 10 test-runs, a convincing result given this particular learning task! C4.5, on the contrary, always produced a more or less complex set of (probably spurious) rules.

- QUIN. This artificial data set was designed to model a task where only probabilistic classification is possible and which also contains disjunctions. It is effectively the same as Quinlan's 'Prob-Disj' data-set [Quinlan 87] and consists of ten random binary variables (v1 to v10). The outcome (Y or N) for each case is assigned according to the conditional expression:

  ```
  IF v1 & v2 & v3
  OR v4 & v5 & v6
  OR v7 & v8 & v9
  THEN outcome = Y (with prob 0.9), outcome = N (prob 0.1)
  ELSE outcome = N (with prob 0.9), outcome = Y (prob 0.1)
  ```

  Attribute v10 is irrelevant. Training set size was 400, test set size was 200.

  Table 1 shows the average accuracy of KNOPF on this data set; it is significantly better than C4.5 at the 95% level. The average accuracy of around 90% is easily explained by the fact that KNOPF in 9 of 10 runs induced exactly the definition given above, the best a non-probabilistic classifier can hope to achieve. Rule sets induced by C4.5 typically consist of the correct rules (or specializations thereof) and some additional spurious rules, thus clearly showing a bit of overfitting.

| | TREEMIN | C4.5 | KNOPF | signif. |
|---|---|---|---|---|
| QUIN | 83.50 | 86.70 | **90.30** | 95% |
| DIGIDAT | **88.32** | 87.79 | 88.26 | no |

Table 1: QUIN and DIGIDAT: Accuracies for KNOPF, C4.5, and TREEMIN, plus t-test significance.

- DIGIDAT. This example is essentially the same as used in [Breiman et al. 84] as a running example. Each data record is generated by simulating a faulty liquid crystal display in which digits are displayed by setting bars on or off. There are seven bars, each of which may be on or off. Training set size was 359, test set size was 642. Every bar had a 0.1 probability of being in error - either on when it should have been off or vice versa. In addition, four spurious attributes randomly set to on or off were included into every example. The learning task was to distinguish 8s and 9s from the other numerals 0 to 7.

  Judging from classification accuracy KNOPF did slightly better than C4.5 on this example and slightly worse than TREEMIN, but the differences are not statistically significant. Judging results qualitatively KNOPF induced just one or two rules except for one test run and these induced rules did not test spurious variables (v8 to v11) for 7 out of the 10 test runs. C4.5 mostly induced three or four rules for the class (plus an equal number of rules for the counter-examples) and also *always* tested at least one spurious variable. This does not seem to hurt in terms of classification accuracy, though. TREEMIN induced trees with 7 leaves, which would correspond to a set of seven rules.

Two other artificial boolean functions, DNF3 and DNF4, introduced in [Pagallo & Haussler 90] turned out to be a hard problem for decision tree learning even though training set sizes are large - 1650 and 2650 respectively. C4.5's performance in this domain is significantly worse than that of KNOPF, which performs rather well even in the presence of noise as can be seen in table 2.

## 5.2 Illegal King-Rook-King Chess Positions

This domain is used as a standard testbed in ILP for experiments involving various amounts of noise and varying sizes of training sets. It is also easily transformed into a propositional learning problem. The standard translation is to rewrite each example into a boolean vector containing 18 variables which represent the 18 different test predicate instantiations that can be used for constructing clauses.

| | Noise | C4.5 | KNOPF | signif. |
|---|---|---|---|---|
| DNF3 | 0.0 | 97.81 | **100.00** | 99% |
| | 0.1 | 88.13 | **98.86** | 99% |
| | 0.2 | 71.83 | **91.39** | 99% |
| DNF4 | 0.0 | 87.25 | **100.00** | 99% |
| | 0.1 | 68.39 | **96.88** | 99% |
| | 0.2 | 61.90 | **91.03** | 99% |

Table 2: DNF3 and DNF4: Accuracies for C4.5 and KNOPF plus t-test significance for various noise levels.

We compare KNOPF, C4.5, and GOLEM, which all do fairly well in this domain.[4] Table 3 summarizes experimental runs. We varied noise levels from 0% to 40% and used training set sizes between 100 and 10000. Testing was always performed on a set of 5000 test examples. Results for KNOPF and C4.5 are averages of 10 runs each. The results reported for GOLEM are as given in [Srinivasan et al. 92] and are probably results of only one run. Furthermore GOLEM had to construct new predicates, too. Empty entries represent the fact that no compression was achieved, i.e. the empty theory resulted from induction.

Comparing the different programs we see that KNOPF almost always induces some theory, and on average these theories are more accurate than the theories induced by the other two programs. Though the differences might not appear to be large, one has to keep in mind the following: a correct and complete theory consists of at least 8 rules, whereas good approximate theories with 4 rules using a total of only 6 tests already yield more than 98% accuracy [Fuernkranz 93].

KNOPF almost always induces some of these approximations, but is also able to construct a complete and correct theory for all ten runs given 10000 examples and misses such a theory in only 1 out of 10 runs given 5000 examples (at noise level 0). Results reported here are also better than those given in [Fuernkranz & Widmer 94] for the case of 10% noise.

Generally C4.5 again induces significantly more rules in all cases, thus producing overly complex theories. This typically does not really hurt performance (but see the above remark about slight differences of accuracy in this domain),

[4]This may be surprising as one would not expect such good results when given at most 10000 out of $2^{18} = 262144$ possible examples for learning. As a matter of fact, these 18 tests form a kind of powerful abstraction language yielding only less than 1000 positive and negative examples each. This can be explained by a lot of interdependecies between the single tests (e.g. $a < b$ and $b < c$ together imply $a < c$). So the original high number of possible board positions, which accidentally also is 262144 (six variables with eight possible values each: $8^6 = 262144$), is reduced by two orders of magnitude. Fortunately, this abstraction also preserves all the information necessary for inducing correct and complete definitions for the *illegal*-concept.

| Noise | Program | 100 | 250 | 500 | 1000 | 5000 | 10000 |
|---|---|---|---|---|---|---|---|
| 0.0 | C4.5 | 98.38 | 98.92 | 99.44 | 99.56 | 99.98 | **100.00** |
| | GOLEM | - | **99.70** | **99.70** | **99.70** | 99.70 | **100.00** |
| | KNOPF | **98.39** | 99.35 | 99.46 | 99.50 | **99.99** | **100.00** |
| 0.05 | C4.5 | 95.74 | **98.87** | 99.00 | 99.39 | 99.65 | **99.96** |
| | GOLEM | - | 98.10 | 98.10 | **99.70** | **99.70** | 99.70 |
| | KNOPF | **96.86** | 98.42 | **99.48** | 99.54 | 99.54 | 99.54 |
| 0.1 | C4.5 | 94.42 | 97.16 | 97.47 | 99.07 | 99.65 | **99.84** |
| | GOLEM | - | - | 98.10 | 98.10 | **99.70** | 99.70 |
| | KNOPF | **96.37** | **98.20** | **99.02** | **99.54** | 99.54 | 99.54 |
| 0.15 | C4.5 | 93.36 | 94.70 | 98.35 | 98.48 | 99.35 | **99.71** |
| | GOLEM | - | - | 98.10 | 98.10 | **99.70** | 99.70 |
| | KNOPF | **96.28** | **98.20** | **98.92** | **99.48** | 99.54 | 99.54 |
| 0.2 | C4.5 | 91.47 | 92.09 | 95.24 | 96.59 | 98.69 | 99.61 |
| | GOLEM | - | - | - | 98.10 | **99.70** | **99.70** |
| | KNOPF | **92.77** | **96.14** | **97.86** | **99.41** | 99.54 | 99.54 |
| 0.3 | C4.5 | **78.71** | **81.85** | 87.80 | 91.37 | 97.08 | 98.10 |
| | GOLEM | - | - | - | - | 98.10 | 98.10 |
| | KNOPF | 77.31 | 77.05 | **88.86** | **97.43** | **99.40** | **99.54** |
| 0.4 | C4.5 | 60.77 | 53.22 | **69.38** | 81.53 | 88.61 | 93.48 |
| | GOLEM | - | - | - | - | - | 98.10 |
| | KNOPF | **67.24** | - | - | **83.89** | **97.59** | **99.23** |

Table 3: Illegal KRK: Accuracies (percentages) for C4.5, GOLEM, and KNOPF for various levels of noise and various sizes of the training set in the King-Rook-King domain. Best results are boldface, except when below base-line accuracy. Empty entries represent null theories which could be attributed base-line accuracy.

though when confronted with small training sets and high noise levels, results may be below the base-line accuracy (which is 66.38% for the particular test set). An example for such a situation is the 250 examples at 40% noise case. Also, results for higher noise levels and larger numbers of training examples exemplify the noise fitting behaviour of C4.5 when compared to GOLEM or KNOPF. Whereas GOLEM prefers to stay agnostic until given enough examples and then produces a high-quality theory, KNOPF produces reasonable theories for all but 2 of the higher noise cases and almost always outperforms the other two approaches at higher noise levels. Furthermore KNOPF never returns a theory yielding less than the base-line accuracy.

# 6 Conclusions, Related Work, and Further Research

We have defined a new MDL measure for rule sets and incorporated it into the inductive learner KNOPF. This new measure is information-theoretically plausible in the way it encodes the theory and the examples and it also gives good results in the experiments reported above. But there are still a lot of open questions and opportunities for improvement.

- Though our way of encoding rules and there estimating their cost seems to produce sensible results empirically, it is definitely not an optimal encoding. Tests are not independent, so after choosing one test, some of the remaining tests are either redundant or contradictory. A better encoding should take this into account.
- The search strategy employed in KNOPF is very greedy, especially regarding the stopping criteria. Maybe a deeper lookahead search could improve overall theories when the current greedy search cannot find compressive rules. The Parity problem seems to be a useful testbed in this regard, as the current version of KNOPF does not fare well in that domain.
- Another open question is the generalization of the proposed MDL measure to account for numbers and new variables. That would make it applicable to all kinds of propositional as well as first-order learning problems.
- Furthermore we want to evaluate the cost of additional attributes induced by means of constructive induction [Bloedorn et al. 93, Pfahringer 94], which can help the learning process in cases of an inadequate initial representation language.

In summary, the new MDL measure proposed in this paper is a generalization of the formula given in [Forsyth 93] applicable to sets of rules, it overcomes the deficiences of the formula used in C4.5, and it is simpler (and may

also be more reliable for small training sets) than the coding scheme used by [Muggleton et al. 92].

## Acknowledgements

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 Gerhard Widmer for constructive discussion and help with this paper, and Johannes Fürnkranz for always collecting relevant literature, discussing all kind of spurious questions, and for providing the king-rook-king position generator.

# References

[Angluin & Laird 87] Angluin D., Laird P.: Learning from Noisy Examples, Machine Learning, 2(4), 343-370, 1987.

[Bloedorn et al. 93] Bloedorn E., Wnek J., Michalski R.S.: Multistrategy Constructive Induction: AQ17-MCI, in Michalski R.S. and Tecuci G.(eds.), Proceedings of the Second International Workshop on Multistrategy Learning (MSL-93), Harpers Ferry, W.VA., pp.188-206, 1993.

[Breiman et al. 84] Breiman L., Friedman J.H., Olshen R.A., Stone C.J.: Classification and Regression Trees, Wadsworth International Group, Belmont, CA, The Wadsworth Statistics/Probability Series, 1984.

[Cheeseman 90] Cheeseman P.: On Finding the Most Probable Model, in Shrager J., Langley P.(eds.): *Computational Models of Discovery and Theory Formation*, Morgan Kaufmann, Los Altos, CA, 1990.

[Forsyth 93] Forsyth R.S.: Overfitting Revisited: An Information-Theoretic Approach to Simpifying Discrimination Trees, in JETAI 6(3), 1994.

[Fuernkranz 93] Fuernkranz J.: A numerical analysis of the KRK domain. Working Note, 1993. Available upon request.

[Fuernkranz & Widmer 94] Fuernkranz J., Widmer G.: Incremental Reduced Error Pruning, Proceedings of the 11th International Conference on Machine Learning (ML-94), New Brunswick, N.J., 1994.

[Lavrac et al. 92] Lavrac N., Cestnik B., Dzeroski S.: Search heuristics in empirical Inductive Logic Programming, in Workshop W18, Logical Approaches to Machine Learning, ECAI-92, Vienna, 1992

[Muggleton et al. 92] Muggleton S., Srinivasan A., Bain M.: Compression, Significance, and Accuracy, in Sleeman D. and Edwards P.(eds.), Machine Learning: Proceedings of the Ninth International Workshop (ML92), Morgan Kaufmann, San Mateo, CA, pp.338-347, 1992.

[Pagallo & Haussler 90] Pagallo G., Haussler D.: Boolean Feature Discovery in Empirical Learning, Machine Learning, 5(1), 71-100, 1990.

[Pfahringer 94] Pfahringer B.: CiPF 2.0: A Robust Constructive Induction System, Proceedings of the Workshop on Constructive Induction and Change of Representation, 11th International Conference on Machine Learning (ML-94/COLT-94), New Brunswick, New Jersey., 1994.

[Quinlan 87] Quinlan J.R.: Simplifying Decision Trees, International Journal of Man-Machine Studies, 27, pp. 221-234, 1987.

[Quinlan & Rivest 89] Quinlan J.R, Rivest R.L.: Inferring Decision Trees using the Minimum Description Length Principle, in Information and Computation, 80:227-248, 1989.

[Quinlan & Cameron-Jones 93] Quinlan J.R., Cameron-Jones R.M.: FOIL: A Midterm Report, in Brazdil P.B.(ed.), Machine Learning: ECML-93, Springer, Berlin, pp.3-20, 1993.

[Quinlan 93] Quinlan J.R.: C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA, 1993.

[Quinlan 94] Quinlan J.R.: The Minimum Description Length Principle and Categorical Theories, in Cohen W.W. and Hirsh H.(eds.), Machine Learning: Proceedings of the Eleventh International Conference (ML94), Morgan Kaufmann, San Mateo, CA, 1994.

[Rissanen 78] Rissanen J.: Modeling by Shortest Data Description, in Automatica, 14:465-471, 1978.

[Srinivasan et al. 92] Srinivasan A., Muggleton S., Bain M.: Distinguishing Exceptions from Noise in Non-Monotonic Learning, in Proceedings of the 2nd International Workshop on Inductive Logic Programming (ILP), Tokyo, 1992.

[Wallace & Patrick 93] Wallace C.S., Patrick J.D.: Coding Decision Trees, Machine Learning, 11(1), 1993.