Controlling Constructive Induction in CIPF: An MDL Approach

Bernhard Pfahringer

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

Abstract. We describe the propositional learning system CIPF, which tightly couples a simple concept learner with a sophisticated constructive induction component. It is described in terms of a *generic architecture* for constructive induction. We focus on the problem of controlling the abundance of opportunities for constructively adding new attributes. In CIPF the so-called *Minimum Description Length* (MDL) principle acts as a powerful control heuristic. This is also confirmed in the experiments reported.

1 Introduction

In learning concept descriptions from preclassified examples, simple concept learners typically make strong assumptions about the way these examples are represented. For effectively learning a concept its examples must populate one or a few regions of the hypothesis space expressible in the description language. For example, decision trees encode axis-parallel nested hyper-rectangles. Two different problems may cause irregular distributions of learning examples in the original representation space: *noise* and/or an *inadequate description language*. Both phenomena lead to complex, convoluted induced concept descriptions which will be hard to understand and will perform poorly at predicting concept membership of unclassified examples.

As a remedy for the latter problem constructive induction has been introduced, e.g. in [Dietterich & Michalski 81] and [Mehra et al. 89]. The basic idea is to somehow transform the original representation space into a space where the learning examples exhibit (more) regularities. Usually this is done by introducing new attributes and forgetting old ones. So constructive induction is searching for an adequate representation language for the learning task at hand.

In this paper we report on CIPF, a generic constructive induction system, and on how search in the representation space is controlled in CIPF. Section 2 briefly describes a generic architecture for constructive induction and discussed CIPF in these terms. In section 3 will focus on how the problem of controlling search for useful representation changes is solved in CIPF by means of the powerful *Minimum Description Length (MDL) Principle* [Rissanen 78]. Section

4 reports experiments and compares results to C4.5 [Quinlan 93], a well-known sophisticated decision tree learner. Section 5 summarizes related work, gives conclusions and talks about further research directions we are pursuing within CIPF.

2 A Generic Architecture and an Introduction to CIPF

This section will briefly describe a generic architecture for constructive induction and use this architecture to introduce CIPF. We will also discuss some important design rationales of CIPF.

Figure 1 depicts a possible generic architecture for describing constructive learners. Most implemented systems can be described in terms of this architecture or a subset of it, if one supplies proper instantiations for the different processes (boxes in the model). The three different processes working together are:

- The CI module: Given examples and attribute descriptions and possibly already some descriptions/hypotheses, this module constructs new attributes according to some methodology. Output of this module are new attribute descriptions and the augmented and transformed learning examples.
- The Selective Learner: Any (classical) propositional learning algorithm can be used to induce rules from the transformed learning data. Output of this module is a set of rules forming a hypothesis that compresses and explains the learning data.
- The Evaluator: This current hypothesis must be evaluated in some way to decide whether it is of good enough quality to serve as a final result, or if it should be input into another cycle of induction. It might also be the case that no good hypothesis is found, but computation nonetheless terminates due to exhausted resources like maximal number of cycles or heuristically/statistically based doubt about the possibility of finding any better hypothesis.

Actual systems not only differ in their choices for the different parameters (e.g. which methods they select for doing CI or what algorithm lies at the heart of their respective learner), they may even omit modules and/or pathways at all; for instance, some systems do not run in cycles, but perform sequential one-shot learning only.

The main goal in building CIPF is designing a practical system for constructive induction that minimizes the number of user-settable parameters. So we try to identify principled choices or automated ways of choosing good values for necessary decisions where other systems rely on user-specified parameter values. This was one reason for choosing the *Minimum Description Length Principle* as an evaluator. This will be described in more detail in the next section.

CIPF borrows heavily from existing systems in that we have tried to collect useful features of known machine learning systems. We try to combine these in a synergetic fashion in CIPF. CIPF is a true instance of the generic architecture

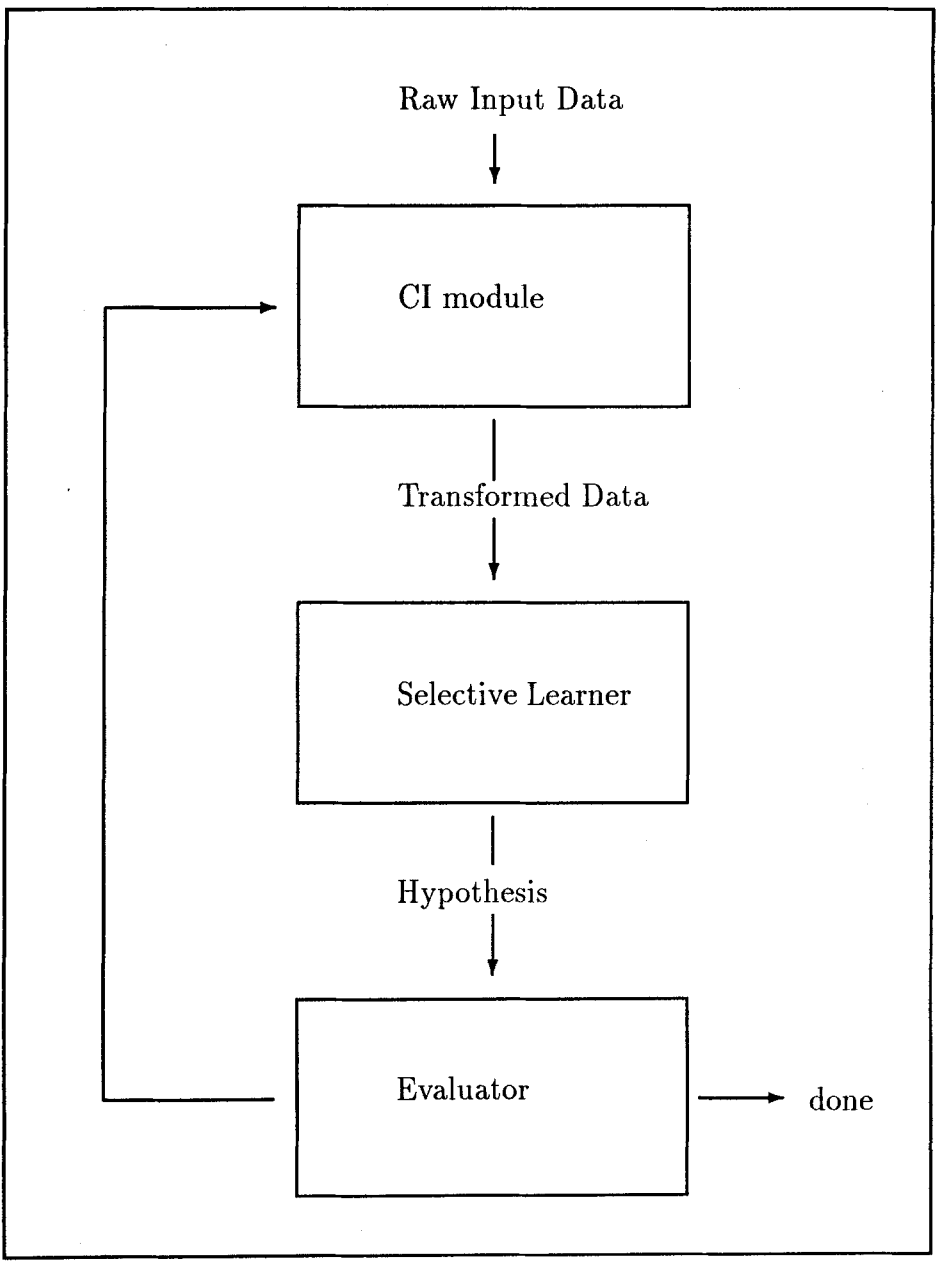

Fig. 1. Constructive induction: a generic architecture

for constructive induction described above in that it realizes all the boxes and pathways. CIPF's components will be detailed in the following.

2.1 Constructive Induction in CIPF (the CI Module)

Just like the multi-strategy system AQ17-MCI [Bloedorn et al. 93], CIPF takes an operator-based approach to constructive induction. It supplies a (still growing) list of generally useful CI operators plus an interface allowing for user-supplied special operators. For instance, these might encode possibly relevant background knowledge. We have currently implemented the following generally useful CI operators in CIPF:

- Compare attributes of the same type: is attribute `A1` *Equal to/Different from/Greater than/Less than* attribute `A2`.
- Discretize numeric attributes into several intervals using Chi-Merge techniques [Kerber 92].
- Conjoin possible values of nominal attributes into sets using modified Chi-Merge as proposed in [Kerber 92].
- Count how many of a given set of boolean attributes are true (or false).
- Conjoin two attributes occuring in a good rule [Matheus & Rendell 89].
- Perform intra-construction [Muggleton 87, Muggleton & Buntine 88] of good rules.
- For the set of positive examples covered by a good rule: compute intervals/subsets for the respective numerical/nominal base-level attributes, so that these intervals/subsets exactly cover these positive examples.
- Drop attributes not used by any of the good rules. [1]
- The medical 3σ-*heuristic*: for numerical attributes construct an attribute testing if this numerical value is in a plausible range. This *healthy* range is operationally defined in terms of mean values and standard deviations derived from the healthy part of a population as the interval $[\mu - 3\sigma, \mu + 3\sigma]$. A value outside such a range is a strong indicator for pathological test results in medical applications [2] (see also section 4.2).

Recursive application of these operators may yield complex new attributes like *the number of numerical attributes being off more than* 3σ *from the 'healthy' mean is zero or one* . It is the user's task to choose the appropriate operators for any learning problem.

2.2 CIPF's Selective Learner

We have implemented a simple propositional FOIL-like learner [Quinlan & Cameron-Jones 93], i.e. our selective learner is a simplified cousin

[1] One might argue whether *dropping an attribute* really is a *constructive induction* operator or not. Anyway it being a very useful operator we have chosen to include it in the above list. Furthermore the terminology used in [Bloedorn et al. 93] defines set of *constructive induction operators* as the union of *constructors* and *destructors*.

[2] Personal communication from a lab physician

of FOIL dealing with propositional horn clauses only. So we are using ideas from *Inductive Logic Programming* and translate them back (specialize them) for propositional problems. We prefer direct induction of rules over decision trees for various reasons. The two most important ones are:

- Unknown values can be dealt with pragmatically: never incorporate tests for *unknown* in a rule.
- Induction focuses on one class at a time. At least in relational learning this approach seems to be superior to decision trees [Watanabe & Rendell 91] and we suspect that the same might be true for propositional learning.

Currently the learner is a quick-and-dirty custom implementation, as we want to focus on constructive induction, but still like to have the possibility of working on the internals of the learner. We will of course have to address the serious shortcomings of this module in further research. Right now this learner in CIPF uses the *Laplace expected error estimate* as a search heuristic, because [Lavrac et al. 92] shows that accuracy estimators outperform information gain criteria when learning rules instead of decision trees. [3] The only stopping criterion used is *no improvement of the estimate*. There is currently no other form of pruning in the learning component.

3 Using MDL to Control Constructive Induction (the Evaluator Module)

CIPF takes a rather eager approach to constructive induction: at every step all possible new attributes are added. This over-abundance in the representation space combined with the simplistic learner quickly results in unwieldy, overly complex induced rule sets when learning without appropriate control. These rule sets may be both difficult to comprehend for the user and yield mediocre results when classifying unseen examples. In analogy to *noise fitting* [Angluin & Laird 87] this phenomenon could be called *language fitting*. Typical examples of such behaviour are published in the section on AQ17-HCI in the *Monk report* [Thrun et al. 91], which describes three artificial learning problems for evaluating and comparing different algorithms. We have made similar experiences with early versions of CIPF lacking sophisticated control.

To prevent CIPF from *language fitting* we have devised the following simple, yet effective control regime:

- Every time the CI module is called, it is allowed to construct an unlimited number of new attributes.
- These attributes will be input to the next learning step. There they will *compete* with each other for being used in induced rules.
- Only the *fittest* attributes will be allowed to survive.

[3] Use of the more general *M-estimate* also discussed in that paper instead of the *Laplace estimate* would introduce one of those parameters we are trying to avoid if possible.

So how are the *fittest* attributes determined in CIPF? We pragmatically equate them with the set of attributes being used by *good* rules. How CIPF determines the set of *good* rules for a class is one of its major innovations. Instead of using some ad-hoc measures of accuracy and quality or some user-supplied evaluation functions we have identified the so-called *Minimum Description Length Principle* [Rissanen 78, Quinlan & Rivest 89] as a very well-performing evaluator.

In a nutshell, MDL is a concept from information theory that takes into account both a theory's simplicity and a theory's predictive accuracy simultaneously. MDL is disarmingly simple: concept membership of each training example is to be communicated from a sender to a receiver. Both know all examples and all attributes used to describe the examples. Now what is being transmitted is a theory (set of rules) describing the concept and, if necessary, explicitly all positive examples not covered by the theory (the false-negative examples) and all negative examples erroneously covered by the theory (the false-positive examples). Now the cost of a transmission is equivalent to the number of bits needed to encode a theory plus its exceptions in a sensible scheme. The MDL Principle states that the best theory derivable from the training data will be the one requiring the minimum number of bits.

So for any set of rules generated by the learner and for subsets of these rules a *cost* can be computed. The rule-set with minimum cost is the best theory for the training data. Only rules of this set will be called *good* rules and will be used as input for the constructive induction module. The precise formula used to apply the MDL Principle in CIPF is the same one as used by C4.5 [Quinlan 93] for simplifying rule sets:

$$Cost = TheoryCost + log_2\left(\binom{C}{FP}\right) + log_2\left(\binom{NC}{FN}\right)$$

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 term 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 number of bits for transmitting a theory and its exceptions.

A slight modification necessary for constructive induction is to take into account also the different complexities of *constructed attributes*. This can easily be achieved in a uniform manner by adding attribute-defining rules to the rule set, one for each *constructed* attribute used in the original rule set. Thus using a *constructed* attribute entails a kind of penalty or cost, which will be amortized either if this attribute offers superior compression or if it is used in more than one rule.

Furthermore, CIPF differs in the way the above MDL estimate is utilized algorithmically. For complexity reasons it is of course impossible to evaluate all possible subsets of rules. [Quinlan 93] reports serious difficulties using greedy hill-climbing and therefore resorts to expensive simulated annealing. In contrast, in our setting a hill-climbing strategy seems to work quite satisfactorily in combination with a preprocessing step as follows:

- Sort all rules induced by the learner in descending order of their estimated accuracy.
- Starting with an empty theory, always add the next best rule to the current theory as long as the MDL estimate improves, i.e. a better compression is achieved.

The output of this algorithm is a subset of all the originally induced rules which will be a good, if not the best theory for the training data in terms of the currently available attributes. Exactly this subset will be used to determine which attributes are to be kept and which are to be dropped for the next cycle of induction: exactly those (original and constructed) attributes are kept which appear in at least one rule of the selected theory. This subset of rules is also used as input for the constructive induction module.

Globally, CIPF does a kind of hill-climbing in the representation space, computing new attributes and new sets of rules utilizing these attributes as long as the overall cost estimate (as measured by the above MDL formula) improves. This last theory is then the overall output of CIPF. Empirically this simple strategy seems to produce good results, as indicated by the experiments reported in the next section and it is effectively computable. Also, to repeat its two main advantages, the strategy includes no user-settable parameters, and also it does not require a secondary training set (*train-test set*), like e.g. AQ17-MCI, to evaluate the quality of constructed attributes.

4 Experiments

In the experiments reported here, CIPF's performance was compared to C4.5 on the same training and test sets. C4.5 is a very sophisticated, production quality selective learner. It was run with default settings and the results reported are for pruned decision trees [4] on the test set.

4.1 Monk's Problems

The *Monk's problems* [Thrun et al. 91] are three artificially constructed problems in a space formed by six nominal attributes having from two to four possible values. These three problems are abbreviated to Monk1, Monk2, and Monk3 in the following. CIPF in its current status gives mixed results for the Monk's

[4] Typically pruned trees yielded better accuracy than both unpruned trees and rule sets generated from the tree.

problems. From table 1 we see that Monk1 was solved without problems. CIPF finds the correct theory:

```
true <= (jacket_color = red)

true <= (head_shape = body_shape)
```

This is no surprise as this example is simple and CIPF has the necessary constructive operator *compare attributes of the same type* at its disposal. C4.5 achieves only 72.4% accuracy for the pruned decision tree, but is able to reach the full 100% for the rules extracted from this tree.

Performance on Monk2 is far from optimal, though. Comparison with C4.5, which is better, but also far from optimal, seems to indicate missing constructive operators. We believe that full negation and disjunction (currently not available in CIPF) may solve Monk2. Alternatively a student at our department is currently finishing work [Kramer 93] on an interesting general constructive operator computing *extensional products* of nominal attributes. This operator seems to be able to solve Monk2 very well.

Results for Monk3 are quite good, but seem to indicate that CIPF has a problem with noise. Potential answers to noise in CIPF will be briefly discussed in the next major section. For C4.5 the unpruned tree is slightly better than the pruned tree, which is not what we expected knowing that the training set exhibits 5% class noise.

To give an impression of both the inferiority of the simple learner currently used in CIPF and the strong abilities of the CI component we have included into table 1 accuracies of the theories induced in the first step (just before the first constructive induction step is taking place). Typically these values are significantly worse than those of C4.5, but with the help of the strong CI component CIPF is able to reach and sometimes even outperform C4.5's predictive performance!

Additionally we would like to mention that some learning system exhibit a much better performance than C4.5 on the *Monk's problems*, e.g. AQ17-HCI achieves 100%, 93.1%, and 100% on Monk1, Monk2, and Monk3 respectively, and a specialized form of Backpropagation yields 100%, 100%, and 97.2% respectively. As already mentioned above we interpret this as an indication for missing constructive operators appropriate for especially the Monk2 problem. [5]

4.2 Two Medical Datasets

For another set of experiments we used two medical datasets, one being the hepatitis data available from the Machine Learning Archive at Irvine, the second being numerical descriptions extracted from cardiac thallium scintigrams recorded at the University of Vienna Medical School [Prem et al. 93]. The first

[5] AQ17-HCI has at its disposal a very special CI operator which perfectly fits the Monk2 problem, thus explaining its impressive performance on this problem.

Table 1. Monk's Problems: accuracies (percentages) for CIPF after the first and the final cycle of induction and for C4.5.

	CIPF first	CIPF final	C4.5
MONK1	70.78	100	72.4
MONK2	66.92	80.7	83.3
MONK3	92.36	97.2	97.2

set exhibits a good mixture of numerical and boolean attributes with a few values missing. The second set is of comparable size (159 examples total), but uses 45 numerical attributes, which we strongly believe to be redundant. The classification task for both datasets is to separate *ill* from *healthy* patients. Experiments were performed with the examples split randomly into equally sized training and test sets for ten runs. Tables 2 and 3 show the respective results of these experiments.

Table 2. Hepatitis data: average number of errors and average accuracy for ten test runs for CIPF and C4.5.

	#Errors	Accuracy
CIPF	14.6	81.29
C4.5	13.3	82.95

Table 3. Scan data: absolute number of errors and their average for ten test runs for CIPF and C4.5.

Run	1	2	3	4	5	6	7	8	9	10	Average #Errors
CIPF	19	15	20	22	16	14	20	20	19	18	18.3
C4.5	23	22	25	22	20	21	29	30	19	22	23.3

The absolute number of errors translate into an average error of 18.71% and 17.05% for CIPF and C4.5 for the hepatitis data, and into 22.87% and 29.13% average error respectively for the scan data. So CIPF performs slightly worse than C4.5 on the first set, but significantly better on the second set. We attribute C4.5's better performance on the hepatitis data to both C4.5's sophisticated handling of noise and to the fact that CIPF's general medical heuristic is not

applicable here. [6] For the scan data, in almost all cases CIPF is significantly better than C4.5. This is a direct consequence of CIPF's constructive abilities. In all these test runs CIPF either constructs an attribute *at most one attribute value is "out of the healthy range"* (see above description of the 3σ*-heuristic)*, which is a good way of characterizing healthy people. Or CIPF constructs the opposite attribute *more than a certain number of attribute values (typically 5) are "out of the healthy range"*, which is well-suited for characterizing people exhibiting serious health problems.

4.3 Inductive Logic Programming Exercises

Encouraged by the original INDUCE system [Dietterich & Michalski 81], which was able to learn *structural* descriptions from examples, and by the current success of LINUS [Dzeroski & Lavrac 91], which essentially translates ILP problems into an attribute-value representation for efficient induction, we started to examine two classical ILP exercises: illegal king-rook-king (KRK) chess endgame positions [Fuernkranz 93] and finite element mesh design [Dolsak & Muggleton 92], [Dzeroski & Bratko 92].

KRK is very easily represented in CIPF. The example tupels of the relation `illegal/6` are transformed into six basic attributes encoding rank and file of all three pieces. Background knowledge in the original formulation consists of definitions for `=/2`, `less_than/2` and `adjacent/2`. Only the last predicate `adjacent/2` had to be encoded as a CI operator, as both `Equal-To` and `Less-Than` are presupplied CI operators in CIPF. Induced theories usually resemble the approximate theories given in [Fuernkranz 93]. A sample theory derived by CIPF from 100 training examples looks as follows:

```
[1] illegal <= (BLACK-KING-FILE = WHITE-ROOK-FILE)

[2] illegal <= (BLACK-KING-RANK = WHITE-ROOK-RANK)

[3] illegal <= (adjacent BLACK-KING-FILE WHITE-KING-FILE) and
               (adjacent BLACK-KING-RANK WHITE-KING-RANK)

[4] illegal <= (adjacent BLACK-KING-FILE WHITE-KING-FILE) and
               (BLACK-KING-RANK = WHITE-KING-RANK)
```

This approximate theory was tested with 5000 test examples yielding an accuracy of 98.4%. This is consistent with [Fuernkranz 93] which proves a theory consisting of the first three clauses 1,2,3 to be 98.451% correct.

For the mesh design domain we did a manual translation along the lines implicitly suggested in [Dolsak & Muggleton 92]. All the one-argument predicates were translated into three nominal attributes with the appropriate sets

[6] Still CIPF discovers regularities missed by C4.5, like that all female patients are healthy. This can be attributed to directly learning rules instead of decision trees.

of possible values. Ignoring all two-argument attributes encoding structure (`neighbor/2`, `opposite/2` and `same/2`) CIPF achieves the surprising results shown in table 4 (results for FOIL,MFOIL,and GOLEM were taken from [Dzeroski & Bratko 92]). One sample rule induced covering 22 positive and no

	FOIL	MFOIL	GOLEM	CIPF
A	17	22	17	21
B	5	12	9	13
C	7	9	5	10
D	0	6	11	23
E	5	10	10	26
SUM	34	59	54	93

Table 4. Mesh Design: Number of Correctly Classified Examples.

negative example looks like the following:

```
N=1 <= (LOAD = ONE_SIDE_LOADED or NOT_LOADED) and
       (EDGE-TYPE  = NOT_IMPORTANT)
```

The ability to form appropriate subsets of possible values of an attribute (called *internal value disjunction* in AQ17-derived systems) seems to provide useful contructed attributes for this learning task. So CIPF without any structural information performs almost twice as well as FOIL, MFOIL, or GOLEM. Still even 93 correctly classified examples translate to only 33.5% accuracy. So there probably is a good chance of achieving much better results by means of a more careful analysis of the mesh design problem itself.

For translating and using the complete original specification automatically we will have to encode constructive operators capable of recursively inspecting objects linked to the object in focus and of summarizing properties of these objects. We are currently designing such operators. These would allow constructing attributes like *this node has a neighbor node with the property* (`edge-type = fixed`) or *this node has at least two* `opposite` *nodes*. Naturally the property to be learned - *number of finite elements for this node* in the mesh domain - could also be represented as an attribute of the example available for inspection by constructive operators. Thus for the mesh domain attributes like *number of finite elements of my* `same` *neighbor* could be constructed which effectively represent a kind of recursive definition. Once such recursive definitions are allowed, additional control will be needed, e.g. to prevent the construction of *cyclic* attributes useless for effective prediction. For instance an attribute *the number of finite elements of my* `same` *neighbor's* `same` *neighbor* would not make sense for prediction, as it references the node in question itself. Pitfalls of recursion in ILP are dealt with at length in [Cameron-Jones & Quinlan 93].

A constructive induction system equipped with such operators might offer an alternative perspective [7] on ILP, possibly providing a more natural fit for data in object-oriented representations or databases.

5 Conclusions, Related Work, and Further Research

Incorporating the MDL Principle into CIPF as the single, uniform heuristic for evaluating theories and thereby implicitly guiding constructive induction proved valuable. The MDL Principle combines both accuracy and complexity of a theory into a single uniform measure. Thus CIPF does not require any ad-hoc measurements or user-defined evaluation functions of possibly questionable quality and can nonetheless use *all* of the available training data for induction. Other approaches (e.g. AQ17-MCI) have to resort to splitting the training data into two or more sub-parts performing some sort of cross-validation on these sub-parts. Such an approach may be more expensive computationally and may miss regularities in the data. Nonetheless, on a systems level, CIPF certainly is most closely related to and influenced by the multi-strategy system AQ17-MCI. The main difference is the way control is imposed on constructive induction. CIPF eagerly tries to use every opportunity for constructive induction until the MDL principle stops this cycling process. AQ17-MCI takes a different approach: relying on a set of meta-rules [Aha 92], it tries to identify the need (*when*) and the directions (*how*) for a change in the representation space. On the operator side AQ17-MCI seems to be more mature especially regarding so-called *deconstructors*. It would certainly be interesting to compare both systems on some tasks using the same set of operators in both systems. A further difference is our aiming at emulating and extending ILP in a constructive induction framework.

Principled Constructive Induction is an interesting concept introduced in [Mehra et al. 89]. Geometric interpretation of the various constructors and the notion of *linear separability* is used to guide the selection of appropriate constructors. These ideas might have interesting implications for CIPF, too.

The problem of *language fitting* is also mentioned and discussed in [Matheus 90] in the context of the CITRE system and a framework for constructive induction. This approach uses additional background knowledge in two different ways when constructing attributes. Domain-knowledge constraints are used to eliminate less desirable new attributes beforehand and domain-dependent transformations generalize newly constructed attributes even further in ways meaningful to the current problem. Though these ideas do not currently fit directly into CIPF's schema for constructive induction, they might still point to valuable further improvements possible for CIPF.

Our further research directions for CIPF include:

- Replacing all other heuristics currently employed by CIPF (e.g. the Laplace estimate as a search heuristic guiding the selective learner) by the MDL principle [Tangkitvanich & Shimura 93]. We have already implemented a simple

[7] at least at the level of implementation

selective learner guided by MDL instead of some accuracy estimator plus stopping criterion. Preliminary experiences seem to suggest robustness regarding noise but a bias towards over-general theories.
- Identifying and implementing additional generally useful constructive operators.
- Improving the selective learner: for instance, the stopping criterion could be modified to take into account the results of the evaluator. From the worst rule still included in the current rule set according to the MDL principle a stronger stopping criterion (like *minimal accuracy*) could be estimated.

An additional endeavour is the search for learning problems at the right level of difficulty. Unfortunately, most of the public machine learning databases at Irvine seem to be easy [Holte 93]. Therefore the best one can hope for for a system like CIPF (and other constructive learners) is to be on a par with sophisticated selective learners (e.g. C4.5) for such databases. We are looking for more difficult and complex learning tasks (tasks where it is hard to define an adequate representation language beforehand), which will allow constructive induction systems to really show their abilities.

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 Gerhard Widmer for constructive discussion and help with this paper.

References

[Aha 92] Aha D.W.: Generalizing from Case Studies: A Case Study, in Sleeman D. and Edwards P.(eds.), Machine Learning: Proceedings of the Ninth International Workshop (ML92), Morgan Kaufmann, San Mateo, CA, pp.1-10, 1992.

[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.

[Cameron-Jones & Quinlan 93] Cameron-Jones R.M., Quinlan J.R.: Avoiding Pitfalls When Learning Recursive Theories, in Bajcsy R.(ed.), Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, pp.1050 -1057, 1993.

[Dietterich & Michalski 81] Dietterich T.G., Michalski R.S.: Inductive Learning of Structural Descriptions: Evaluation Criteria and Comparative Review of Selected Methods, Artificial Intelligence, 16(3), 257-294, 1981.

[Dolsak & Muggleton 92] Dolsak B., Muggleton S.: The Application of Inductive Logic Programming to Finite-Element Mesh Design, in Muggleton S., Inductive Logic Programming, Academic Press, London, U.K., 1992.

[Dzeroski & Lavrac 91] Dzeroski S., Lavrac N.: Learning Relations from Noisy Examples: An Empirical Comparison of LINUS and FOIL, in Birnbaum L.A. and Collins G.C.(eds.), Machine Learning: Proceedings of the Eighth International Workshop (ML91), Morgan Kaufmann, San Mateo, CA, pp.399-402, 1991.

[Dzeroski & Bratko 92] Dzeroski S., Bratko I.: Handling Noise in Inductive Logic Programming, Proceedings of the 2nd International Workshop on Inductive Logic Programming, 1992.

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

[Holte 93] Holte R.C.: Very Simple Classification Rules Perform Well on Most Commonly Used Datasets, Machine Learning, 11(1), 1993.

[Kerber 92] Kerber R.: ChiMerge: Discretization of Numeric Attributes, in Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI Press/MIT Press, Menlo Park, pp.123-128, 1992.

[Kramer 93] Kramer S.: CN2-MCI: Ein zweistufiges Verfahren für konstruktive Induktion, Master's thesis in preparation, Vienna, 1993.

[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

[Matheus & Rendell 89] Matheus C.J., Rendell L.A.: Constructive Induction On Decision Trees, in Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89), Morgan Kaufmann, Los Altos, CA, 645-650, 1989.

[Matheus 90] Matheus C.J.: Adding Domain Knowledge to SBL Through Feature Construction, in Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI -90), AAAI Press/MIT Press, Menlo Park, CA, pp.803-808, 1990.

[Mehra et al. 89] Mehra P., Rendell L.A., Wah B.W.: Principled Constructive Induction, in Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89), Morgan Kaufmann, Los Altos, CA, 651-656, 1989.

[Muggleton 87] Muggleton S.: Duce, An Oracle-based Approach to Constructive Induction, in Proceedings of the 10th International Joint Conference on Artificial Intelligence (IJCAI-87), Morgan Kaufmann, Los Altos, CA, p.287-292, 1987.

[Muggleton & Buntine 88] Muggleton S., Buntine W.: Machine Invention of First Order Predicates by Inverting Resolution, in Laird J.(ed.), Proceedings of the Fifth International Conference on Machine Learning, Univ.of Michigan, Ann Arbor, June 12-14, Morgan Kaufmann, San Mateo, CA, pp.339-352, 1988.

[Prem et al. 93] Prem E., Mackinger M., Dorffner G., Porenta G., Sochor H.: Concept Support as a Method for Programming Neural Networks with Symbolic Knowledge, in Ohlbach H.J.(ed.), GWAI-92: Advances in Artificial Intelligence, Springer, Berlin, Lecture Notes in AI, Vol.671, 1993.

[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.

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

[Tangkitvanich & Shimura 93] Tangkitvanich S., Shimura M.: Learning from an Approximate Theory and Noisy Examples, in Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI -93), AAAI Press/MIT Press, Menlo Park, CA, pp.466-471, 1993.

[Thrun et al. 91] Thrun S.B., et.al.: The MONK's Problems: A Performance Comparison of Different Learning Algorithms, CMU Tech Report, CMU-CS-91-197, 1991.

[Watanabe & Rendell 91] Watanabe L., Rendell L.: Learning Structural Decision Trees from Examples, in Proceedings of the 12th International Conference on Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, pp.770-776, 1991.