GalaII: Integrating the Construction of Boolean and Prototypical Features

Yuh-Jyh Hu and Dennis Kibler
Information and Computer Science Department
University of California, Irvine
{yhu, kibler}@ics.uci.edu

Abstract

We examine a new approach to constructive induction that combines Boolean feature construction with prototypical feature construction. This particular representation language appears to be useful in many naturally occurring domains. Despite its representational power, the results are reasonably comprehensible. Also, the learning process is efficient. We present evidence for the effectiveness of this approach on real and artificial domains. Through a series of systematic experiments, we partially characterize the generality of the approach.

1 Introduction

There is no best learning algorithm over all possible domains. However natural domains, i.e. domains that have been collected by people, reflect a hidden bias. We anticipate/hope that these domains have comprehensible concept descriptions. Nonetheless the initial or raw features that are used to describe such domains may not reveal the putative regularities. One approach to mitigate this problem is to construct new features. The need for useful new features has been suggested by many researchers (Matheus, 1991; Ragavan *et. al.*, 1993). Constructing new features by hand is often difficult (Quinlan 1993). The goal of constructive induction is to automatically generate

new features so that the regularity is more apparent (Dietterich & Michalski, 1981), thus yielding improved classification accuracy. The constructed features should make concept learning easier for many learning algorithms.

There are currently many constructive induction algorithms based on the strategy of constructing new attributes, including FRINGE (Pagallo, 1989), GREEDY3 (Pagallo & Haussler, 1990), DCFringe (Yang *et. al.*, 1991), CITRE (Matheus & Rendell, 1989), LFC (Ragavan & Rendell, 1993; Ragavan *et. al.*, 1993), MRP (Perez & Rendell 1995), GALA (Hu & Kibler 1996), etc. Unfortunately, most of the current constructive induction algorithms degrade in performance as the target concept becomes larger and more complex, when measured by the number of Boolean combinations (Perez & Rendell 1996). Though MRP is demonstrated to learn several complex relations, the meaning of its extensional representation is implicit and usually difficult to interpret.

Constructive induction has been directed towards improved classification accuracy. Since natural domains have been selected because of putative hidden relations, we believe that biasing constructive induction to find features that are reasonable comprehensible will be a useful heuristic, i.e. one that matches the bias that occurs in natural domains. To put it more forcefully we maintain:

> Hypothesis: Natural Domains are biased towards comprehensible descriptions.

Consequently we bias our constructive algorithm towards comprehensible concepts. In our previous work on GALA we demonstrated how one could find effectively find additional Boolean combinations of features by a process of selective iterative lookahead search. These combinations were typically small but outside the search horizon of standard hill-climbing algorithms. We maintain that these derived features are relatively easy to comprehend. In this paper we introduce a new type of feature, which we call a Simple Linear Feature (SLF), which are also reasonable comprehensible. SLFs are extensions of k-of-m concepts but not as general as Linear Threshold Units. These concepts allow one to express in a comprehensible manner concepts which would be unwieldy in a Boolean formulation. Moreover these type of features appear to occur in many natural domains, such as medical domains (Sprackman 1988) In this paper we describe GalaII, which combines

construction of SLFs with the construction of Boolean features via GALA. We demonstrate that GalaII generates features that are useful for several different learning algorithms on several domains. We also obtain a partial characterization of GalaII, via empirical studies on artificial domains.

2 Motivation

One important issue in constructive induction is the types of new attributes constructed (Hu & Kibler 1996). If we represent the new attributes in disjunctive normal form, we could build a spectrum of the attributes based on the number of terms involved. At one end are the features with relatively few terms; at the other end, those with many terms. We aim at covering those portions of the spectrum that appear to occur naturally. GALA is an approach to constructive induction which uses selective lookahead to generating complex but relatively small Boolean features (Hu & Kibler 1996). This covers one end of the spectrum.

As concepts becomes larger in terms of the number of relevant attributes involved, selective lookahead algorithms such as GALA fail to find useful new attributes. Larger numbers of attribute combinations incur more attribute interaction, which hinders the performance (Hu & Kibler 1996; Perez & Rendell 1996). Therefore, we propose another construction strategy to cover this end of the spectrum. Because it is impossible to learn any arbitrarily long and complex concept, we concentrate on finding prototypical concepts, i.e. concepts which depend on the accumulative value of many primitive features.

We carry this process out this integration in a several stages. First we train a neural network on the concept. The hidden units from the neural net are trasformed into SLFs. A Simple Linear Feature is one with small integer values for its weights. The SLF captures regularities that are impossible to find using GALA. Last, we use GALA to find new Boolean combinations of the primitive and SLF features. We begin by describing the process to construct the SLF from a hidden unit.

3 Constructing Simple Linear Features

In general, the learned hidden units from a neural network can be viewed as a constructive induction. However these units are usually not easilty interpretable. Here we describe the goals and process by which we create a STF from a hidden unit.

We maintain that the concepts such as k-of-m are easily understood. We aim to approximate a hidden unit by a linear threshold unit which has few non-zero integer weights. Such units we call Simple Linear Features (SLFs). In general, it is not possible to maintain fidelity to the original Hidden Unit with this limited representation. However our experience with natural domains indicates that this approach is effective.

The algorithm has two main components. In one component we normalize and round the size of the weights by dividing by the mean of the average absolute values. This removes some of the complexity due to the unnecessary use of real numbers and yields a more intuitive description of the relative importance of the features. For example, 3-of {X1,X2,...X5} is more comprehensible than $1.02X_1 + 0.98X_2 + 1.05X_3 + 0.97X_4 + 0.98X_5 \geq 2.93$. This process only slightly changes the predictions of a node. Another component removes features which are contrary to the primary decision of the node and of small value. This attempts to remove irrelevant features which typically have non-zero weights in the hidden unit. The removal of these features also requires an adjustment of the threshold. This adjustment is the most expensive part of the algorithm and requires time linearly proportional to the number of examples and the number of features. The details of the algorithm are given in figure 1. This process retains those features whose cumulative value may be important.

To illustrate this process and its effect, consider the concept $(\sum_{i=1}^{8} x_i \geq 5) + x_1\bar{x}_3\bar{x}_5x_7 + \bar{x}_2\bar{x}_4\bar{x}_6x_8$, a Boolean function based on 12 attributes with four irrelevant attributes (x_9 ... x_{12}). Using 5% of the total 4096 examples as the training set, we trained a neural net with the backpropagation algorithm. The neural net has 12 input units, 6 hidden units and one output unit. A subset of the weights and thresholds are provided in figure 4. At this point the network is difficult to interpret, even when we know what the net should represent.

In figure 3, we provide the results of the SLF construction process. We display only the hidden units 1 and 5, but the others are similar. In the neural

```
Given : Neuron with weights W and threshold T
        a set of training examples E
Return: Simple Linear Threshold Function

 Normalize weights:
   replace weight by weight/(mean of absolute weight values)
 Remove  attributes:
    Set weight to 0 if its weight is of opposite sign
    to largest weight and is less than Mean,
 Normalize weights
 Adjust Thresholds:
  Add to threshold the sum of differences of predictions over E
    before and after changes to weights
 Round weights to nearest integer
```

Figure 1: SLF Construction

net, the weights and thresholds learned are not understandable even though the fifth hidden unit approximately converges to $\sum_{i=1}^{8} x_i \geq 5$. However, after SLF construction, the fifth hidden unit is easily interpreted as a 5-of-8 rule. Once these SLFs are added to the representation language, our previous program GALA is used to learn Boolean combinations.

3.1 Related Work

The most closely related work is the research on rule-extraction from neural networks. The main goal in rule-extraction is to rerepresent the neural net as a set of comprehensible rules while maintaining fidelity to the original net. Our goal in constructive induction is different. We glean from the net useful features which will aid any learning algorithm in generating a concept description. Consequently it is not surprising the methods employed are different. There are several search-based approaches for extracting conjunctive rules and m-of-n rules from neural networks (Fu, 1991; Cravan & Shavlik 1993). They either conduct a breadth-first search for possible com-

```
Weights of hidden units before normalization :

Hidden Unit 1:  4.43948,-0.90867,-1.85273,
               -0.09461,-2.56182, 0.44131,
                3.80673, 0.88315, 1.21998,
               -0.02205, 0.03032,-1.60084,
                threshold = -7.18849

Hidden Unit 5:  3.13984, 3.82628, 3.94562,
                3.75279, 3.37203, 3.75229,
                3.07394, 3.34460,-0.24504,
                0.06971,-0.26181,-0.70992,
                threshold = -15.6727


Weights of hidden units after normalization :

Hidden Unit 1:  2, 0,-1, 0,-1, 0, 1, 0, 0,
                0, 0,-1,   threshold = -2

Hidden Unit 5:  1, 1, 1, 1, 1, 1, 1, 1, 0,
                0, 0, 0,   threshold = -5
```

Figure 2: Results of Normalization

binations of conjunctive rules or extract m-of-n rules by encouraging weight clustering during training through special training methods. These approach tend to be computationally expensive, while in our approach the major cost is training the neural network. The process of SLF construction has linear time-complexity in terms of the number of input features and training examples.

Another interesting approach towards generating rules from neural nets is the TREPAN program (Craven & Shavlik 1994, 1996). Here the neural network is regarded as a black-box or oracle which can be queried. In this way one has an unlimited number of virtual examples to supply to any rule-learning program. It is necessary to be careful to maintain the same distribution of examples, else the virtual examples will mislead the learning program. Our approach, which is not aimed at making the entire neural network comprehensible, does not incur this sampling difficulty.

4 Experimental Results on Real Domains

In these experiments our goal is to demonstrate that combining the SLF construction with construction of Boolean features is effective. We only use natural domains, i.e. domains that have been collected from data and not artificially generated. In particular we use: breast cancer, heart disease, liver disorder, ionosphere, lymphography and promoters. In each case, two thirds of the examples form the training set and the remaining examples form the test set. For comparison with other feature learning programs, we also used LFC when it was applicable. LFC combines its generation of new attributes with its concept learning program. The results are shown in table 1. Each result is averaged over 20 runs. The second through fourth column is the accuracy of the neural net, C4.5 and LFC respectively. Because LFC is a two-class learning algorithm, we did not apply it to lymphography. The fifth (NN+B) and six columns (C4.5+B) are the new accuracies of the neural net and C4.5 after adding the new Boolean attributes generated by GALA. The seventh column (C4.5+S) is the new accuracy of C4.5 after adding the new SLF attributes. The last column (C4.5+B+S) is the new accuracy of C4.5 with the addition of both the Boolean and SLF features.

In all experiments, the parameters of C4.5 were set to the default values. For the neural net, the learning rate and the momentum term were both

Table 1: Results of Real Domains

Domain	NN	C4.5	LFC	NN+B	C4.5+B	C4.5+S	C4.5+B+S
Breast	92.7	93.6	94.2	95.1	95.2	95.1	95.2
Heart	74.3	72.3	75.2	76.9	76.4	74.5	76.7
Liver	66.1	62.1	62.4	68.7	65.4	62.3	65.6
Ionosphere	87.3	90.1	90.3	90.3	92.2	90.1	92.3
Lymphography	87.1	73.9	N/A	87.3	79.1	77.7	78.1
Promoter	71.9	73.9	75.1	77.1	79.5	78.5	82.0

set to 0.1, and the attribute values were normalized between 0 and 1. We also adopted the suggested heuristics for a fully connected network structure: initial weights selected at random and a single hidden layer whose number of nodes was half the total number of input and output nodes (Ragavan & Rendell, 1993; Rumelhart *et. al.*, 1986).

Although this is a small set of domains, several patterns can be clearly seen. First, the addition of Boolean features to both the neural net and C4.5 improved learning. Second, the additional SLF features to C4.5 usually improved performance. Third, the addition of both Boolean and SLF features to C4.5 improved performance of C4.5 in all cases and achieved a higher accuracy than either NN or C4.5 in 4 out of 6 domains. Over this collection of domains, the addition of the SLF features yielding insignificant improvement in 3 domains, insignificant degradation in 1 domain, and significant improvement in only one domain. In the next section we provide evidence that the process of constructing SLFs is both effective and useful in increasing accuracy. We demonstrate this by considering artificial domain which contain prototypical subconcepts.

5 Experimental Results on Artificial Domains

The following set of experiments are designed to partially characterize the kind of features that can be constructed. First we demonstrate that if the domain contains a "pure" prototypical concept, then the SLF construction process will generate the appropriate description. Second we consider concepts that are hybrids of Boolean features and prototypes. Here we distinguish two cases: those domains where the prototype dominates and those where the Boolean features dominate. In this more difficult task we demon-

strate that the integrated approach used by GalaII can determine both the prototypical and Boolean features. We also demonstrate that the resulting derived features improves classification accuracy for both C4.5 and backpropagation, providing evidence that the derived features could not be found by the learning algorithm themselves.

In the following experiments, each Boolean function is defined on 12 Boolean attributes (some may be irrelevant), yielding a population of 4096 examples. We used 5% and 10% of the examples for training all the rest for testing.

5.1 Pure Prototypical Concepts

The first experiments verifies that the SLF process finds the pure prototypical concepts. We tested several prototypical concepts, including m-of-n, majority, exact-m-of-n, etc. The entire set of functions tested are described in table 2. The results are averaged over 20 runs, and reported in table 3.

The second (and sixth) column of table 3 denotes the percentage of times averaged over 20 runs that SLF process successfully extracted the prototype. By this we mean that the SLF process generated the feature which was identical to the linear threshold feature that used in generating the Boolean concept. The third and fourth (also seventh and eighth) columns denote the accuracy of the neural net and C4.5 (Quinlan 1993) respectively. The new accuracy of C4.5 after adding the new SLF is presented in column 5 and 9.

With only 5% of the data, GalaII was usually able to find the prototype, except for the difficult concept P4. With 10%, GalaII was usually able to exactly extract the hidden linear threshold function. Obviously, the additional features were extremely useful for C4.5.

5.2 Hybrid Concepts

By a hybrid concept we mean one that is a combination of both a prototypical feature and Boolean features, for example $(\sum_{i=1}^{8} x_i \geq 5) + x_1\bar{x}_3\bar{x}_5x_7 + \bar{x}_2\bar{x}_4\bar{x}_6x_8$. Concepts of this form provide a more difficult test for extracting either the Simple Linear Features or the Boolean Features.

We consider two types of hybrid concepts. One is where the linear threshold unit covers significantly more examples than any of the Boolean terms.

Table 2: Summary of Pure Prototypical Concepts

Concept	Description
P1	$\sum_{i=1}^{12} x_i \geq 5$
P2	$(\sum_{i=1}^{6} x_i \geq 4) + (\sum_{i=7}^{12} x_i \geq 4)$
P3	$(\sum_{i=1}^{6} x_i \geq 4) + (\sum_{i=5}^{10} x_i \geq 4)$
P4	$\sum_{i=1}^{10} x_i = 3$
P5	$\sum_{i=2}^{10} x_i \geq 5$
P6	$\sum_{i=5}^{6} x_i = \sum_{i=7}^{8} x_i$
P7	$\sum_{i=3}^{6} x_i = \sum_{i=7}^{10} x_i$
P8	$\sum_{i=3}^{6} x_i > \sum_{i=7}^{10} x_i$

Table 3: Results of Pure Prototypical Concepts

	5%				10%			
Concept	%	NN	C4.5	C4.5+P	%	NN	C4.5	C4.5+P
P1	100	99.9	82.2	100.0*	100	100.0	82.7	100.0*
P2	95	98.8	73.4	98.9*	100	99.9	77.5	100.0*
P3	65	97.5	79.2	96.1*	95	99.9	84.7	99.7*
P4	15	93.4	83.2	94.3*	85	98.6	83.9	99.2*
P5	100	100.0	76.4	100.0*	100	100.0	79.9	100.0*
P6	100	99.4	72.1	100.0*	100	100.0	77.3	100.0*
P7	100	99.5	63.8	100.0*	100	99.9	64.9	100.0*
P8	100	100.0	79.9	100.0*	100	100.0	84.3	100.0*

Table 4: Summary of Dominant Prototypical Concepts

Concept	Description
D1	$(\sum_{i=1}^{8} x_i \geq 5) + x_1\bar{x}_3\bar{x}_5x_7 + \bar{x}_2\bar{x}_4\bar{x}_6x_8$
D2	$(\sum_{i=1}^{8} x_i \geq 5) + x_9\bar{x}_{11}\bar{x}_1x_3 + \bar{x}_{10}x_{12}x_2\bar{x}_4 + x_{11}\bar{x}_1x_3x_5$
D3	$(\sum_{i=1}^{8} x_i \geq 5) + x_9\bar{x}_{10}\bar{x}_{11}x_{12} + \bar{x}_9x_{10}\bar{x}_{11}\bar{x}_{12} + x_9\bar{x}_{10}x_{11}\bar{x}_{12}$
D4	$(\sum_{i=1}^{6} x_i \geq 3) + x_1\bar{x}_2\bar{x}_3x_4 + \bar{x}_2x_3\bar{x}_4\bar{x}_5 + x_3\bar{x}_4x_5\bar{x}_6$
D5	$(\sum_{i=1}^{6} x_i \geq 3) + x_1\bar{x}_3\bar{x}_7x_8 + \bar{x}_3x_4\bar{x}_8x_9 + x_2\bar{x}_5x_9x_{10}$
D6	$(\sum_{i=1}^{6} x_i \geq 3) + x_7\bar{x}_8\bar{x}_9x_{10} + \bar{x}_9x_{10}\bar{x}_{11}\bar{x}_{12} + x_7\bar{x}_9x_{10}\bar{x}_{11}$
D7	$(\sum_{i=1}^{8} x_i \geq 5) + x_1\bar{x}_2\bar{x}_3 + \bar{x}_3x_5\bar{x}_8 + \bar{x}_4x_6x_7 + x_7x_8\bar{x}_4$
D8	$(\sum_{i=1}^{8} x_i \geq 5) + x_1\bar{x}_9\bar{x}_{10} + \bar{x}_3x_{11}\bar{x}_{12} + \bar{x}_5x_7\bar{x}_9 + x_6\bar{x}_{10}x_{11}$
D9	$(\sum_{i=1}^{8} x_i \geq 5) + x_9\bar{x}_{10}x_{11} + x_9x_{11}\bar{x}_{12} + x_{10}\bar{x}_{11}\bar{x}_{12} + x_9x_{11}x_{12}$

We call this type "dominate prototypical concepts". Otherwise we call the concept a "non-dominate prototypical concept".

5.2.1 Dominant Prototypical Concepts

In this category, we tested 9 Boolean concepts summarized in table 4. The literals in the Boolean combinations could be included in, separated from, or overlapped with the features of the prototype.

The results are reported in table 5. The second (and ninth) column indicate when GalaII extracts the identical linear threshold function, a very severe test. We provide two measures for evaluating GalaII's ability to extract Boolean features. One measure (the first number in the third and tenth columns) presents the average number of terms constructed by GalaII that correspond exactly to the Boolean term in the target concept. The second measure (the second number in the third and tenth column) counts the number of terms that GalaII approximates. We say a conjunction T' approximates a conjunction T if it is the conjunction of terms of T with at least two terms and is not identical to the term. For example, the term x_1x_2 approximates $x_1x_2x_3$.

The fourth and fifth (also eleventh and twelfth) columns denotes the accuracy of the neural net and C4.5 respectively. The new accuracy of C4.5 after adding the new attributes generated by the neural net alone is presented

Table 5: Results of Dominant Prototypical Concepts

5%								10%						
Concept	%	DNF	NN	C4.5	C4.5+P	C4.5+B	C4.5+P+B	%	DNF	NN	C4.5	C4.5+P	C4.5+B	C4.5+P+B
D1	85	1.1,0.0	94.4	78.4	94.7	82.2	95.5	100	1.2,0.2	99.3	82.2	99.5	88.8	99.7
D2	55	0.8,0.3	90.5	77.7	87.3	81.3	91.0	90	2.0,0.4	96.3	80.7	96.1	85.7	98.2
D3	50	1.0,0.3	92.7	68.6	86.2	81.3	91.8	85	1.0,0.8	99.7	72.7	97.3	85.2	97.9
D4	50	0.0,1.1	96.9	91.5	97.2	93.0	97.4	40	0.0,1.0	99.6	97.1	99.1	98.5	99.2
D5	70	0.1,0.6	94.5	85.1	93.1	86.8	93.8	85	0.7,0.6	96.5	91.5	95.8	92.6	97.8
D6	100	0.0,1.0	97.1	84.3	96.1	86.0	97.3	100	0.0,1.7	98.5	89.7	98.1	92.9	99.2
D7	15	0.4,0.1	88.8	82.2	86.1	84.3	89.3	35	0.9,0.1	95.7	88.6	93.8	91.4	94.2
D8	15	0.3,0.3	85.1	75.1	82.2	80.1	84.0	70	1.8,0.2	92.9	81.1	93.4	85.2	95.5
D9	80	0.1,0.8	95.5	78.8	95.9	85.2	97.5	100	0.4,0.7	99.8	82.2	99.9	89.6	99.9

Table 6: Summary of Nondominant Prototypical Concepts

Concept	Description
ND1	$(\sum_{i=1}^{5} x_i \geq 4) + x_1 x_3 \bar{x}_4 + x_2 \bar{x}_3 x_5 + x_1 \bar{x}_2 \bar{x}_4 + x_2 x_4 \bar{x}_5$
ND2	$(\sum_{i=1}^{5} x_i \geq 4) + x_1 \bar{x}_9 \bar{x}_{10} + \bar{x}_3 x_{11} \bar{x}_{12} + \bar{x}_5 x_7 \bar{x}_9 + x_6 \bar{x}_{10} x_{11}$
ND3	$(\sum_{i=1}^{5} x_i \geq 4) + x_6 \bar{x}_8 \bar{x}_9 + x_8 \bar{x}_{10} x_{11} + x_7 \bar{x}_9 x_{12} + x_7 x_{11} \bar{x}_{12}$

in column 6 and 13. The new accuracy of C4.5 with the addition of the new attributes generated by GALA is shown in column 7 and 14. Column 8 and 15 denote the new accuracy of C4.5 after adding all the new attributes constructed by the neural net and GALA together.

5.2.2 Nondominant Prototypical Concepts

For the second category, we tested 3 concepts. These concepts are summarized in table 6, and table 7 presents the results. As the data indicates, for this form of hybrid concept the SLF construction component of GalaII is unable to correctly tease out the prototypical structure. Consequently the addition of these features potentially degrades the performance of C4.5. Nonetheless, the Boolean feature generator component of GalaII finds useful new features so that the performance of C4.5 is dramatically improved for some concepts.

Table 7: Results of Nondominant Prototypical Concepts

	5%							10%						
Concept	%	DNF	NN	C4.5	C4.5+P	C4.5+B	C4.5+S+B	%	DNF	NN	C4.5	C4.5+S	C4.5+B	C4.5+S+B
ND1	0	0.3,1.2	97.3	98.4	97.8	99.1	99.1	0	0.0,0.1	99.7	99.5	99.8	100.0	100.0
ND2	15	2.1,0.2	84.3	80.2	82.6	91.5	90.5	45	2.5,0.1	94.2	86.6	94.8	99.0	98.5
ND3	0	2.8,0.0	82.3	82.5	81.1	92.7	92.7	25	2.0,0.5	92.5	88.5	93.2	99.2	96.1

5.3 Analysis

In the first experiment, all the Boolean functions are either linear threshold concepts or simple combinations of linear threshold concepts. As anticipated, the SLF generation successfully extracted the prototype in almost all cases. This demonstrates that when the target concept is a pure prototypical one, the SLF construction correctly identifies the hidden linear threshold function. By adding these new attributes the accuracy of C4.5 is significantly improved.

When the target concept is hybrid of prototypes and Boolean terms, the performance of SLF construction depended on whether the prototype was dominant or not. When the prototype dominated the hybrid concept, SLF construction usually identified the underlying linear threshold function. Once the prototypical structure was correctly identified, then the Boolean construction portion of GalaII could correctly identify the remaining Boolean terms.

When the prototype does not dominate the hybrid concepts, SLF construction fails. In this case, treats the whole target concept as a large Boolean combination concept, and extracts as many terms as possible. These terms may be either part of the prototypical structures or of the Boolean combinations.

6 Conclusion

The main objective of this paper was to extend our previous constructive induction algorithm to cover a larger and more useful portion of the space of possible Boolean function. Towards this end we introduced the notion of Simple Linear Features, which generalizes k-of-m concepts but is more restrictive than general linear threshold functions. These features also seem

to be reasonably comprehensible. We also provided an efficient algorithm for learning SLFs, one that is linear in the number of features and examples (not counting the major cost of building a neural net). We also showed how to combine this construction with GALA, a program that generates Boolean features. The result is that all the features generated are reasonable understandable.

We demonstrated the effectiveness of GalaII on a half-dozen real domains, showing that with the new features both backpropagation and C4.5 learned more accurate descriptions.

Last we used synthetic data to characterize the type of features that GalaII is capable of discovering. In particular we showed that if the concept had a dominate prototype component, then SLF construction would generally correctly identify this part of the concept. In this case, GalaII would significant increase the accuracy of C4.5. If the prototypical structure was not dominate, then SLF failed to find the prototype and any increase in accuracy for C4.5 was due to the Boolean feature construction component.

Our overall goal is to aim constructive induction at natural domains where we expect that comprehensible features will be most valuable. In the future we will examine real-valued attributes in natural domains to discover the type of relationships that occur among them. Then we will bias a constructive induction algorithm to find these type of relationships.

References

Craven, M. W. & Shavlik, J. W. "Learning Symbolic Rules Using Artificial Neural Networks", in Proceeding of the 10th International Conference on Machine learning, p73-80, 1993.

Craven, M. W. & Shavlik, J. W. "Extracting Tree-Structured Representations of Trained Networks", in Advances in Neural Information Processing Systems, Vol 8, MIT Press, Cambridge, MA, 1996.

Dietterich, T. G. & Michalski, R. S. "Inductive Learning of Structural Description : Evaluation Criteria and Comparative Review of Selected Methods", Artificial Intelligence 16 (3), p257-294, 1981.

Fu, L. "Rule Learning by Searching on Adapted Nets", in Proceeding of the 9th National Conference on Artificial Intelligence, p590-595.

Hu, Y. & Kibler, D. "Generation of Attributes for Learning Algorithms",

in Proceeding of the 13th National Conference on Artificial Intelligence, p806-811, 1996.

Kung, S. Y. "Digital Neural Networks", Prentice Hall, 1993.

Matheus, C. J. & Rendell, L. A. "Constructive Induction on Decision Trees", in Proceeding of the 11th International Joint Conference on Artificial Intelligence, p645-650, 1989.

Pagallo, G. "Learning DNF by Decision Trees", in Proceeding of the 11th International Joint Conference on Artificial Intelligence.

Perez, E. & Rendell, L. "Using Multidimensional Projection to Find Relations", in Proceeding of the 12th Machine Learning Conference, p447-455, 1995.

Perez, E. & Rendell, L. "Learning Despite Concept Variation by Finding Structure in Attribute-based Date", in Proceeding of the 13th Machine Learning Conference, p391-399, 1996.

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

Ragavan, H., Rendell, L., Shaw, M., Tessmer, A. "Complex Concept Acquisition through Directed Search and Feature Caching", in Proceeding of the 13th International Joint Conference on Artificial Intelligence, p946-958, 1993.

Ragavan, H. & Rendell, L. "Lookahead Feature Construction for Learning Hard Concepts", in Proceeding of the 10th Machine Learning Conference, p252-259, 1993.

Rumelhart, D.E., Hinton, G. E., Williams, R. J. "Learning Internal Representations by Error Propagation" in Parallel Distributed Processing: Explorations in Microstructures of Cognition, Vol 1, p318-362, 1986.

Spackman, K. "Learning Categorical Decision Criteria in Biomedical Domains", in Proceeding of the 5th International Workshop on Machine Learning, p36-46, 1988.