A Wrapper Approach for Constructive Induction

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

Abstract

Inductive algorithms rely strongly on their representational biases. Representational inadequacy can be mitigated by constructive induction. This paper introduces the notion of a relative gain measure and describes a new constructive induction algorithm (GALA) which is independent of the learning algorithm. GALA generates a small number of new boolean attributes from existing boolean, nominal or real-valued attributes. Unlike most previous research on constructive induction, our methods are designed as preprocessing step before standard machine learning algorithms are applied. We present results which demonstrate the effectiveness of GALA on both artificial and real domains for both symbolic and subsymbolic learners. For symbolic learners, we used C4.5 and CN2. For subsymbolic learners, we used perceptron and backpropagation. In all cases, the GALA preprocessor increased the performance of the learning algorithm.

A Wrapper Approach for Constructive Induction

Word Count: 5252 ID: A278 : Keywords: Machine Learning

Submissions: A different version of this paper will be submitted to the Machine Learning Conference.

Abstract

Inductive algorithms rely strongly on their representational biases. Representational inadequacy can be mitigated by constructive induction. This paper introduces the notion of a relative gain measure and describes a new constructive induction algorithm (GALA) which is independent of the learning algorithm. GALA generates a small number of new boolean attributes from existing boolean, nominal or real-valued attributes. Unlike most previous research on constructive induction, our methods are designed as preprocessing step before standard machine learning algorithms are applied. We present results which demonstrate the effectiveness of GALA on both artificial and real domains for both symbolic and subsymbolic learners. For symbolic learners, we used C4.5 and CN2. For subsymbolic learners, we used perceptron and backpropagation. In all cases, the GALA preprocessor increased the performance of the learning algorithm.

1 Introduction

The ability of an inductive learning algorithm to find an accurate concept description depends heavily upon the representation (Flann & Dietterich, 1986). Concept learners typically make strong assumptions about the vocabulary used to represent these examples. The vocabulary of features determines not only the form and size of the final concept learned, but also the speed of the convergence (Fawcett & Utgoff, 1991). Learning algorithms that consider a single attribute at a time may overlook the significance of combining features. For example, C4.5 (Quinlan, 1993) splits on the test of a single attribute while constructing a decision tree, and CN2 (Clark & Niblett, 1989; Clark & Boswell,1991) specializes the complexes in Star by conjoining a single literal or dropping a disjunctive element in its selector.

Such algorithms suffer from the standard problem of any hill-climbing search: the best local decision may not lead to the best global result.

One approach to mitigate these problems is to construct new features. The need for useful new features has been suggested by many researchers (Matheus, 1991; Aha, 1991; Kadie, 1991; Ragavan *et. al.*, 1993). Constructing new feature by hand is often difficult (Quinlan, 1983). The goal of constructive induction is to automatically transform the original representation space into a new one where the regularity is more apparent (Dietterich & Michalski, 1981; Mehra et. al., 1989), thus yielding improved classification accuracy.

Several machine learning algorithms perform feature construction by extending greedy, hill-climbing strategies, including FRINGE (Pagallo, 1989), GREEDY3 (Pagallo & Haussler, 1990), DCFringe (Yang et. al., 1991), CITRE (Matheus & Rendell, 1989). These algorithms construct new features by finding local patterns in the previously constructed decision tree. These new attributes are added to the original attributes, the learning algorithm is called again, and the process continues until no new useful attributes can be found. However, such greedy feature construction algorithms often show their improvement only on particular artificial concepts (Yang et. al., 1991). FRINGE-like algorithms are limited by the quality of the original decision tree from which they construct new features. Some reports show that if the basis for the construction of the original tree is a greedy, hill-climbing strategy, accuracies remain low (Rendell & Ragavan, 1993).

An alternative approach is to use some form of lookahead search. Exhaustive lookahead algorithms like IDX (Norton, 1989) can improve decision tree inductive learners without constructing new attributes, but the lookahead is computationally prohibited, except in simple domains. LFC (Ragavan & Rendell, 1993; Ragavan et. al., 1993) mitigate this problem by using directed lookahead and by caching features. Search complexity limits this approach also. In the next section, we introduce a new feature construction algorithm which addresses both the bias of the initial representation and the search complexity.

2 Issues

There are several important issues about the constructive induction process.

Interleaved vs Wrapper

By interleaved we mean that the learning process and constructive induction process are intertwined into a single algorithm. Most current constructive induction algorithms fall into this category. This limits the applicability of the constructive induction method. By keeping these processes separate, as in a wrapper model, the constructive induction algorithm can be used as a preprocessor to any learning algorithm. With the wrapper model one can also test the appropriateness of the learned features over a wider class of learning algorithms. GALA follows the wrapper approach.

Hypothesis-driven vs. Data-driven

Hypothesis-driven approaches construct new attributes based on the hypotheses generated previously. This is a two-edged sword. It has the advantage of taking advantage of previous knowledge and the disadvantage of being strongly dependent on the quality of previous knowledge. GALA is data-driven.

Absolute Measures and Relative Measures

Absolute measures evaluate the quality of an attribute on the training examples without regard to how the attribute was constructed. Examples of such measures include entropy, gain ratio, or error-rate. While it is important that new attributes perform well, it is also important that they are different from the ones that would have been discovered by the learning algorithm. To measure this quality we consider the improvement of a constructed attribute's quality over the quality of its parents. We refer to this as a relative measure since it depends on the quality of its parents. GALA uses both relative and absolute measures, and, may be the only constructive induction algorithm to include a relative measure.

Number of new attributes

It has been reported that the performance learning algorithms degrade when given spurious new attributes. This follows from the general result: the larger the hypothesis space, the less likely the learning algorithm can correctly identify the concept with the same sample size. Consequently we

prefer constructive induction algorithms that generate few features. GALA generates a small number of useful features.

Operators

The simplest operators for constructing new attributes are boolean operators, which are what most constructive induction algorithms use. One could also consider relational operators or operations based on clustering. GALA only uses (iteratively) the boolean operators "and" and "not".

Domain of Applicability

By this we mean the possible types of attributes that the constructive induction algorithm can use. Some constructive inductions algorithms only deal with boolean attributes. Constructive inductive algorithms could deal with attributes defined by first-order formual or real-valued functions. GALA applies to attributes that are boolean, nominal, or real-valued.

3 The GALA Algorithm

GALA (Generation of Attributes for Learning Algorithms) extends previous work on feature construction by allowing the primitive features to be nominal and real-valued as well as boolean. Roughly, the idea is to consider those constructed features which have high relative and absolute gain ratio. This will be defined more precisely. In later sections we show the advantage of GALA. We also show the value of using a combined metric with ablation studies.

The algorithm has four basic steps. The general flow of the algorithm is generate and test, but it is complicated since testing is interlaced with generation, to keep down the computational cost.

The basic processes of the algorithm are:

Booleanize: Two boolean attributes are associated with each nominal and real-valued attribute.

Generation: Conjunctions and negations of boolean attributes are generated. Since the process is iterated, any boolean feature can be generated.

Table 1: GALA

```
Given: Primitive attributes P and training examples E
Return: a set of new attributes NEW

Procedure GALA(P,E)
 Initialize NEW to be empty.
 If (size(E) > threshold)  and  (E is not all of same class)
  Then Set Bool to boolean attributes from Booleanize(P,E)
       Set Pool to attributes from Generate(Bool,E)
       Set Best to attribute in Pool with highest gain ratio
         (if more than one, pick one of smallest size).
       Add Best to NEW.
       Split on Best and call GALA on each partition
  Else return NEW.
```

Gain Filtering: This filter uses absolute gain ratio and relative gain ratio to decide on which attributes to consider.

The overall control flow is given in Table 1. The filtering routines are not apparent since they are called from within the generation process. For each partition of the data, only one new attribute is added to the original set of primitives. Partitioning is stopped when the set is homogeneous or below a fixed size, currently set at 10. The following subsections describe the basic steps in more detail.

3.1 Booleanize

Suppose f is a feature and v is any value in its range. We define two boolean attributes, Pos(f,v) and Neg(f,v) as follows:

$$Pos(f,v) \equiv \begin{cases} f = v & \text{if f is a nominal or boolean attribute} \\ f > v & \text{if f is a continuous attribute} \end{cases}$$

Table 2: Transforming real and nominal attributes to boolean attributes

```
Given:  Attributes P and examples E.
Return: set of candidate boolean attributes.

Procedure Booleanize(P,E)
   Set Bool to empty.
   For each attribute f, find the v
       such that Pos(f,v) has highest gain ratio on E.
       Add Pos(f,v) and Neg(f,v) to Bool.
   Return Bool
```

$$Neg(f,v) \equiv \begin{cases} f \neq v & \text{if f is a nominal or boolean attribute} \\ f \leq v & \text{if f is a continuous attribute} \end{cases}$$

The idea is to transform each real or nominal attribute to a single boolean attribute choosing a single value from the range. The algorithm is more precisely defined in table 2. This process takes O(AVE) time where A is the number of attributes, V is the maximum number of attribute-values, and E is the number of examples. The net effect is that there will be two boolean attributes associated with each primitive attribute.

3.2 Generation

Conceptually, GALA uses only two operators, conjunction and negation, to construct new boolean attributes from old ones. Repetition of these operators yields the possibility of generating any boolean feature, however only attributes with high heuristic value will be kept. Table 3 describes the iterative and interlaced generate and test procedure. Each cycle of the generate-and-test procedure takes O(A^2E) time where A is the number of attributes and E is the number of examples.

Table 3: Attribute Generation

```
Given: a set of boolean attributes  P
       and a set of  training examples E
Return: new boolean attributes

Procedure Generate(P,E)
  Let Pool be P.
  Repeat
     For each conjunction of Pos(f,v) or Neg(f,v)
         with Pos(g,w) or Neg(g,w).
       If conjunction passes GainFilter, add it to Pool.
  Until no new attributes are found
  Return Pool
```

Table 4: Filtering by mean absolute and relative gain ratio

```
Given: a set of Attributes N
Return: new attributes with high GR and RGR

Procedure GainFilter(N)
  Set M to those attributes in N whose gain ratio is
      better than mean(GR(N)).
  Set M' to those attributes in M whose UPPER_RGR is
      better than mean(UPPER_RGR(N)) or LOWER_RGR is
      better than mean(LOWER_RGR(N))
  Return M'.
```

3.3 Heuristic Gain Filtering

In general, if A is an attribute we define GR(A) as the gain ratio of A. If a new attribute A is the conjunction of attributes A1 and A2, then we define two relative gain ratios associated with A as:

$$UPPER_RGR(A) = max\{\frac{GR(A) - GR(A1)}{GR(A)}, \frac{GR(A) - GR(A2)}{GR(A)}\}.$$

$$LOWER_RGR(A) = min\{\frac{GR(A) - GR(A1)}{GR(A)}, \frac{GR(A) - GR(A2)}{GR(A)}\}.$$

We only consider the relative gain ratio when the conjunction has a better gain ratio than each of its parents. Consequently this measure ranges from 0 to 1 and is a measure of the synergy of the conjunction over the value of the individual attributes. To consider every new attribute during feature construction is computational impractical. We believe that coupling relative gain ratio with gain ratio constrains the search space without overlooking many useful attributes. We define mean(GR(S)) as the average absolute gain ratio of each attribute in S. We also define the mean relative gain ratios (mean(UPPER_RGR(S)) and mean(LOWER_RGR(S))) over a set S of attributes similarly. We use these measures to define the gain-filtering described in table 4.

4 Experimental Results

We carried out three types of experiments. First, we show that GALA performs comparably with LFC on an artificial set of problems that Ragaven & Rendell(1993) used. They have kindly provides us with the code to LFC so that we could also perform other tests. Second, we consider the performance of GALA on a number of real world domains. Last, we consider various ablations studies to verify that our system is not overly complicated, i.e. the various components add power to the algorithm.

4.1 Boolean Function Experiments

We chose the same four boolean function as did Ragavan & Rendell (1993) and used their training methodology. Each boolean was defined over 9 attributes, where 4 or 5 attributes were irrelevant. The training set had 32 examples and the remaining 480 were used for testing. LFC is an interleaved constructive induction algorithm, so we can only report it improvement over C4.5. Since GALA follows a wrapper model we can also report the effects on other learning algorithms. The four boolean functions were:

$$f_1 = x_1x_2x_3 + x_1x_2x_4 + x_1x_2x_5$$
$$f_2 = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_2x_4\bar{x}_3 + \bar{x}_3\bar{x}_4\bar{x}_1$$
$$f_3 = \bar{x}_5\bar{x}_3x_6 + \bar{x}_6x_8\bar{x}_5 + x_8\bar{x}_3\bar{x}_2$$
$$f_4 = \bar{x}_6x_1x_8 + x_8x_4\bar{x}_1 + \bar{x}_9\bar{x}_8x_1$$

By examining these boolean functions it is easy to see that the functions get progressively more difficult to learn, as verified by the experiments with C4.5.

The results of this experiment with respect to the learning algorithms C4.5, CN2(using Laplace accuracy instead of entropy), perceptron, and backprop are reported in table 5. Since GALA generates boolean features, it is not suprising that perceptron learning would be greatly inhanced for these concepts. It is surprising that GALA+perceptron is better than C4.5 on the three most difficult functions and achieved the best performance of any algorithm on f_3 and f_4. It is important to note that GALA always significantly improved backpropagation and usually improved the performance of C4.5 and CN2, all of which have the capability to learn any feature that GALA proposed. This demonstrates that the construction process is different than the learning one. Also it shows that the features generated by GALA are valuable with respect to several rather different learning algorithms. We believe this is a result of combining relative and absolute measure, so that the constructed attribute will be different from the ones generated by the learning algorithm.

Previous reports indicated that LFC outperfoms many other learners in several domains (Ragavan & Rendell, 1993; Ragavan et. al., 1993). The results of LFC are also reported in the table. In table 5 from top-to-bottom the rows denote: the accuracy of the learning algorithm, the accuracy of the learning algorithm using the generated attributes, the concept size (node

Table 5: Accuracy and Hypothesis Complexity Comparison in artificial domains. Significant improvement by GALA over learning algorithms marked with $*$, and significant difference between GALA+C4.5(or CN2,perceptron,backprop) and LFC marked with 1(or 2,3,4).

Function	f_1	f_2	f_3	f_4
C4.5	95.1 ± 3.2	90.2 ± 3.4	80.2 ± 5.2	74.9 ± 8.7
GALA+C4.5	95.6 ± 2.0	$95.2 \pm 5.7^*$	$87.9 \pm 6.8^*$	$85.9 \pm 7.8^*$
size	5.6	7.8	9.3	10.6
size+	3.0^*	3.0^*	3.0^*	3.0^*
CN2	93.6 ± 3.3	91.8 ± 5.9	86.8 ± 6.4	83.6 ± 7.9
GALA+CN2	$95.5 \pm 2.2^*$	$95.2 \pm 3.9^*$	87.1 ± 6.0	83.9 ± 7.6
size	5.1	6.1	7.8	7.9
size+	3.2^*	4.2^*	3.8^*	4.1^*
perceptron	83.9 ± 7.4	79.3 ± 4.3	77.3 ± 4.5	66.2 ± 4.1
GALA+perceptron	$92.2 \pm 6.7^*$	$94.7 \pm 3.9^*$	$91.1 \pm 4.4^*$	$84.9 \pm 5.2^*$
backprop	88.3 ± 6.3	84.3 ± 3.7	81.5 ± 3.9	73.3 ± 4.9
GALA+backprop	$93.5 \pm 4.4^*$	$93.1 \pm 3.1^*$	$86.6 \pm 5.4^*$	$84.0 \pm 6.9^*$
LFC	94.2 ± 4.1	93.3 ± 6.9	84.0 ± 6.3^{1234}	82.3 ± 8.1^1
size	3.9	5.5^1	6.9^1	7.1^1
used	1.5	2.2	2.9	3.1

count for C4.5 and number of selectors for CN2) without generated attributes, the concept size after using GALA, the accuracy of LFC, the concept size (node count for LFC), and the number of new attributes LFC used. In all of these experiments, GALA produced an average of only one new attribute and this attribute was always selected by C4.5 and CN2. There was no significant difference between the accuracy of LFC and C4.5+GALA on these concepts.

The differences of hypothesis complexities and accuracies are significant at 0.01 level. Because CN2, which is a rule induction system, is different from decision tree induction algorithms, we did not compare its hypothesis complexity with LFC's. Notice that in no case was the introduction of new attributes detrimental and in 13 out of 16 cases the generated attribute was useful.

4.2 Real Domains

We are more interested in demonstrating the value of GALA on real world domains. The behavior of GALA on these domains is very similar to its behavior on the artificial domains.

We selected several difficult domains which have a mixture of nominal and continuous attributes. The domains used were: Cleveland heart disease, Bupa liver disorder, Credit screening, Pima Diabetes, Wisconsin Breast Cancer, Wine and Promoters. In each case, two thirds of the examples form the training set and the remaining examples form the test set. The results of these experiments are given in table 6. We do not report the results for the Diabetes domain since there was no significant difference for any field. Besides providing the same information as in the artificial domains, we have also added two additional fields, the average number of new attributes generated by GALA and the average number of attributes used (included in the final concept description) by the learning algorithm. Notice we did not apply LFC or perceptron to the wine domain because they require 2 classes and the wine domain has 3 classes. Again each result is averaged over 20 runs.

Table 6 shows the results of accuracies, hypothesis size, number of new attributes added and number of new attributes used. The differences of concept complexities are significant at the 0.01 level, and the differences of accuracies are significant at the 0.02 level. Notice that in no case was the introduction of the generated attributes harmful to the learning algorithm and in 21 out of 27 cases (6 out of 7 for C4.5, 4 out of 7 for CN2, 5 out

Table 6: Accuracy and Hypothesis Complexity Comparison in real domains. Significant improvement by GALA over learning algorithms marked with $*$, and significant difference between GALA+C4.5(or CN2,perceptron,backprop) and LFC marked with 1(or 2,3,4).

Domain	Heart	Liver	Credit	Breast	Wine	Promoter
C4.5	72.3 ± 2.1	62.1 ± 5.0	81.6 ± 2.5	93.6 ± 1.4	89.5 ± 4.9	73.9 ± 8.8
GALA+ C4.5	$76.4 \pm 2.5^{*}$	$65.4 \pm 3.8^{*}$	$83.3 \pm 2.2^{*}$	$95.2 \pm 1.8^{*}$	$93.8 \pm 3.0^{*}$	$79.5 \pm 7.8^{*}$
size	26.7	77.4	117.8	33.6	9.5	24.4
size+	16.9^{*}	73.7^{*}	99.7^{*}	27.3^{*}	6.7^{*}	14.3^{*}
generated	2.1	1.0	2.5	1.0	1.9	3.3
used	1.8	1.0	1.9	1.0	1.4	2.9
CN2	73.8 ± 2.7	65.2 ± 3.1	83.1 ± 2.6	95.1 ± 1.0	91.6 ± 3.7	74.1 ± 8.5
GALA+ CN2	$76.1 \pm 3.2^{*}$	$68.2 \pm 5.5^{*}$	82.1 ± 2.7	94.8 ± 1.5	$93.5 \pm 4.4^{*}$	$78.3 \pm 7.1^{*}$
size	34.8	85.7	99.4	37.8	16.4	21.0
size+	25.3^{*}	80.1^{*}	131.7^{*}	32.2^{*}	11.0^{*}	17.0^{*}
used	1.7	1.0	2.0	1.0	1.7	2.8
perceptron	59.5 ± 7.1	58.4 ± 7.2	73.5 ± 4.2	91.2 ± 3.7	N/A	74.5 ± 5.5
GALA+ perceptron	$71.7 \pm 12.9^{*}$	$64.3 \pm 6.6^{*}$	$78.9 \pm 6.9^{*}$	$93.4 \pm 1.7^{*}$	N/A	$76.7 \pm 4.8^{*}$
backprop	64.4 ± 4.2	66.1 ± 4.1	77.2 ± 7.7	92.7 ± 1.4	85.5 ± 10.1	71.9 ± 6.1
GALA+ backprop	$76.9 \pm 3.2^{*}$	$68.7 \pm 4.2^{*}$	$83.4 \pm 2.7^{*}$	$95.1 \pm 1.6^{*}$	$93.7 \pm 3.9^{*}$	$77.1 \pm 7.4^{*}$
LFC	75.2 ± 2.7^{134}	62.4 ± 4.5^{1234}	79.9 ± 2.4^{124}	94.2 ± 1.4^{14}	N/A	75.1 ± 7.0^{12}
size	18.8^{1}	32.7^{1}	119.9^{1}	33.4^{1}	N/A	7.4^{1}
used	8.5	9.9	49.9	13.6	N/A	3.2

of 6 for perceptron, 6 out of 7 for backprop) GALA significantly increased the resulting accuracy. Excluding the Diabetes domain (on which nothing seem to improve performance) GALA always improved the performance of both backpropagation and perceptron, mimicking the results for the artificial concepts. It also improved C4.5 on all these domains. For CN.2 it improved the performance on 4 out of these six domains and never degraded performance. In two cases where in degraded performance (not significantly) CN.2 was the best performing algorithm. In general we see that GALA improves the performance of the learning algorithms and those that are performing poorly with respect to other algorithms, are improved the most.

Table 7 indicates some sample learned new attributes. This table shows that the complexity of the generated attributes is sometimes high, making them unlikely to be generated by naive lookahead search or by iteratively conjoining features from an initial decision tree. In this table we have presented the generated features in a more readable form, as the algorithm achieves $a \vee b$ via $\sim (\sim a \wedge \sim b)$.

4.3 Ablation Studies

To further demonstrate the value of combining absolute gain ratio with relative gain ratio, we experimented with just using one of these conjuncts. We reran the same experiments using only the relative measure and found that the accuracy dropped dramatically as the concept complexity grows. Due to space consideration, we only report the results on C4.5+GALA and C4.5+$GALA^-$ in table 8, where $GALA^-$ stands for the ablated algorithm. In this case performance, as measured by accuracy, dropped significantly. When we reran the experiments using only the absolute measures, the accuracies were not harmed, but search space increased dramatically by 25% to 200%, varying on domains. Thus, the experimental results show that the combined heuristic constrain the search space without hurting the resulting accuracy.

5 Related Work

FRINGE-like algorithms construct new features from an initial decision tree. Such strategies are insufficient when attributes interact strongly (Rendell & Ragavan, 1993). Unlike FRINGE-like algorithms, GALA is not limited to the

Table 7: Sample of Learned New Attributes

Domain	Sample Learned New Attribute
Heart	$(thalach > 162)(chol > 149) \vee (thalach > 162)(thal \neq normal)$ $\vee(cp = asymptomatic)(ca \neq 0) \vee (slope \neq upsloping)(ca \neq 0)$
Liver	$(alkphos > 23)(sgot > 12)(gammagt > 8)(drinks \leq 12)$
Credit	$(A_3 > 25.125)(A_8 \leq 13.875) \vee (A_3 > 25.125)(A_9 = f)(A_{13} \neq s)$ $\vee(A_4 = y)$
Diabetes	$(Number\ of\ times\ pregnant \leq 13)(Body\ mass\ index \leq 57.3)$ $(Oral\ glucose\ tolerance \leq 161)(Diastolic\ blood\ presure \leq 110)$
Breast	$(Clump\ thickness > 6)(Uniformity\ of\ cell\ shape > 3)\vee$ $(Clump\ thickness > 6)(Bland\ chromatin > 3)\vee$ $(Uniformity\ of\ cell\ shape > 3)(Marginal\ adhesion > 3)\vee$ $(Marginal\ adhesion > 3)(Bland\ chromatin > 3)\vee$ $(Clump\ thickness > 6)(Uniformity\ of\ cell\ size > 3)\vee$ $(Uniformity\ of\ cell\ size > 3)(Bland\ chromatin > 3)\vee$ $(Uniformity\ of\ cell\ shape > 3)(Bland\ chromatin > 3)$
Wine	$(A_1 \leq 12.77)(A_7 > 0.92) \vee (A_1 \leq 12.77)(A_{10} \leq 3.4)\vee$ $(A_5 \leq 88)(A_7 > 0.92) \vee (A_5 \leq 88)(A_{10} \leq 3.4)$
Promoter	(p-36 = t)(p-12 $\neq$ g)

Table 8: Effectiveness Test (C4.5+GALA/C4.5+$GALA^-$)

Function	C4.5+GALA	C4.5+$GALA^-$
f_1	95.6 ± 2.0	93.8 ± 4.4
f_2	95.2 ± 5.7	91.9 ± 5.7
f_3	87.9 ± 6.8	84.0 ± 6.3
f_4	85.9 ± 7.8	77.8 ± 9.1

repeated local patterns in the initial decision tree. In our experiments, most of the new attributes generated by GALA can neither be found in the decision trees produced by C4.5 nor in the decision lists produced by CN2. In contrast, GALA constructs new attributes by iteratively conjoining attributes that have high relative and absolute gains. This directs the algorithm to construct worthwhile and novel features.

Another way to discover attribute interaction is by using lookahead (Rendell & Ragavan, 1993), but naive lookahead search is expensive. LFC (Ragavan & Rendell, 1993, Ragavan et. al. 1993) uses directed lookahead search to avoid computational explosion. However, LFC uses a variant of averaged entropy as its quality measure which may also run into the same danger of overlooking promising attributes as mentioned earlier. Besides, LFC constructs new attributes only from a beam of existing attributes and the original attributes. This method is more efficient than naive lookahead search,but it also restricts the construction of new attributes (Perez & Rendell, 1995). Furthermore, LFC is currently a 2-class learner which cannot be applied to multiple-class domains. In contrast, GALA constructs new features from the entire set of existing attributes, and uses the relative gain measure to direct lookahead search.

6 Conclusion and Future Directions

This paper presents a new approach to constructive induction which generates a small number (1 to 3 on average) of boolean features from real-valued, nominal and boolean features. The GALA method is independent of the learning algorithm so can be used as a preprocessor to various learning algorithm. We demonstrated that it usually improves the performance of CN2 and C4.5 and nearly always improves the performance of perceptron and backpropagation learning. The method relies on combining an absolute measure of quality with a relative one, which encourages the algorithm to find new features that are outside the space generated by standard machine learning algorithms. Using this approach we demonstrated significant improvement in several artificial and real-world domains and no degradation in accuracy in any domain. The usefulness of the generated features was demonstrated for four different learning algorithms, C4.5, CN2, perceptron and backprop.

With hindsight it is not surprising that GALA was so useful for a perceptron. Essentially GALA constructs crisp boolean features which are outside the representational space of a perceptron. Perceptron learning with GALA generated features was nearly as good as more general symbolic learning and was most dramatic when the symbolic learner greatly outperformed percerptron learning. In our future research we want to consider several questions. If a subsymbolic learner outperforms a symbolic one, can we generate new features that will improve the performance of the symbolic learner? Can we demonstrate that these new features will not harmfully interact with a large class of learning algorithms? Can we integrate these methods with those of GALA into a single effective algorithm?

References

Aha, D. "Incremental Constructive Induction: An Instanced-Based Approach", in Proceeding of the 8th Machine Learning Workshop, p117-121, 1991.

Clark, P. & Boswell, R. "Rule Induction with CN2: some recent improvements", European Working Session on Learning, p151-161, 1991.

Clark, P. & Niblett, T. "The CN2 Induction Algorithm", Machine Learning 3, p261-283, 1989.

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.

Fawcett, T. E. & Utgoff, P. E. "Automatic Feature Generation for Problem Solving Systems", in Proceeding of the 9th International Workshop on Machine Learning, p144-153, 1992.

Flann, N. S. & Dietterich, T. G. "Selecting Appropriate Representations for Learning from Examples", in Proceeding of the 5th National Conference on Artificial Intelligence, p460-466, 1986.

Kadie, C. M. "Quantifying the Value of Constructive Induction, Knowledge, and Noise Filtering on Inductive Learning", in Proceeding of the 8th Machine Learning Workshop, p153-157, 1991.

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.

Matheus, C. J. "The Need for Constructive Induction", in Proceeding of the 8th Machine Learning Workshop, p173-177, 1991.

Mehra, P., Rendell, L. A., Wah, B. W. "Principled Constructive Induction", in Proceeding of the 11th International Joint Conference on Artificial Intelligence, p651-656, 1989.

Norton, S. W. "Generating better Decision Trees", in Proceeding of the 11th International Joint Conference on Artificial Intelligence, p800-805, 1989.

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

Pagallo, G. & Haussler, D. "Boolean Feature Discovery in Empirical Learning", Machine Learning 5, p71-99, 1990.

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

Quinlan, J. R. "Learning efficient classification procedures and their application to chess end games" , in Michalski et. al.'s Machine Learning : An artificial intelligence approach. (Eds.) 1983.

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.

Rendell L. A. & Ragavan, H. "Improving the Design of Induction Methods by Analyzing Algorithm Functionality and Data-Based Concept Complexity", in Proceeding of the 13th International Joint Conference on Artificial Intelligence, p952-958, 1993.

Yang, D-S., Rendell, L. A., Blix, G. "A Scheme for Feature Construction and a Comparison of Empirical Methods", in Proceedingof the 12th International Joint Conference on Artificial Intelligence, p699-704, 1991.