Generation of Attributes for Learning Algorithms

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. Constructive induction can mitigate representational inadequacies. 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. Unlike most previous research on constructive induction, our methods are designed as preprocessing step before standard machine learning algorithms are applied. We present the results which demonstrate the effectiveness of GALA on artificial and real domains for several learners: C4.5, CN2, perceptron and backpropagation.

Introduction

The ability of an inductive learning algorithm to find an accurate concept description depends heavily upon the representation. 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. 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; Aha 1991), thus yielding improved classification accuracy. Several machine learning algorithms perform feature construction by extending greedy, hill-climbing strategies, including FRINGE, 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 artificial concepts (Yang *et. al.*, 1991). FRINGE-like algorithms are limited by the quality of the original decision tree. 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. However, LFC's quality measure (i.e., blurring measure) limits this approach.

In this paper, we introduce a new feature construction algorithm which addresses both the bias of the initial representation and the search complexity in constructive induction.

Issues

There are several important issues about the constructive induction process.

Interleaving vs Preprocessing

By interleaving 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, the constructive induction algorithm can be used as a preprocessor to any learning algorithm. With the preprocessor model one can also test the appropriateness of the learned features over a wider class of learning algorithms. GALA follows the preprocessing approach.

Hypothesis-driven vs Data-driven

Hypothesis-driven approaches construct new attributes based on the hypotheses generated previously. This is a two-edged sword. They have the advantage of previous knowledge and the disadvantage of being strongly dependent on the quality of previous knowledge. On the other hand, data-driven approaches cannot benefit from previous hypotheses, but can avoid the strong dependence. GALA is data-driven.

Absolute Measures vs 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 variants (e.g., blurring), gain ratio, and error-rate. While it is important that a new attribute performs well, it is also important that it has significant improvement over its parents. We refer to this as a relative measure. GALA uses both relative and absolute measures. GALA's relative measure is different from STAGGER's (Schlimmer, 1987) in two respects. First, STAGGER metrics are based on statistical measures rather than information measures. Second, STAGGER evaluates the quality of a son by its absolute difference in quality from it parents (i.e., using parent's quality as threshold) while GALA uses the relative quality difference.

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. Currently GALA only uses (iteratively) the boolean operators "and" and "not".

Attribute Types

We say an attribute is *crisp* if it has a relatively small description as a Boolean combination of the primitive features. Otherwise the attribute is not crisp. A common type of non-crisp attributes are prototypical attributes. A prototypical attribute corresponds to a m-of-n concept. For example the 5-of-10 concept requires 252 conjuncts when described in disjunctive normal form. Obviously there is a spectrum between crisp and non-crisp attributes. GALA finds crisp attributes.

The GALA Algorithm

The idea of GALA (Generation of Attributes for Learning Algorithms) 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 three basic steps. The general flow of the algorithm is generate-and-test, but it is complicated since testing is interlaced with generation. The overall control flow is given in Figure 1. 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.

```
Given: Primitive attributes P, training examples E, threshold,
       cycle limit c and new attributes NEW
        (NEW is empty when GALA invoked the first time)
Return: a set of new attributes NEW

Procedure GALA(P,E,threshold,c,NEW)
  If (size(E) greater than 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,c)
         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
         N = empty set
         For each outcome, Si, of Split on Best
           Ei = examples with outcome Si on split
           NEWi = GALA(P,Ei,threshold,c,NEW)
           N = union of N and NEWi
         NEW = union of NEW and N
         Return NEW
    Else Return empty set
```

Figure 1: GALA

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

Procedure Booleanize (P,E)
  Set Bool to empty.
  For each attribute f in P, 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
```

Figure 2: Transforming real and nominal attributes to boolean attributes

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:

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

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

The idea is to transform each real or nominal attribute to a single boolean attribute by choosing a single value from the range. The algorithm is more precisely defined in Figure 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

```
Given: a set of boolean attributes P
        a set of  training examples E
        cycle limit C
Return: new boolean attributes

Procedure Generate(P,E,C)
  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 or reach cycle limit C
  Return Pool
```

Figure 3: Attribute Generation

```
Given: a set of new 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'.
```

Figure 4: Filtering by mean absolute and relative gain ratio

boolean attributes associated with each primitive attribute.

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. Figure 3 describes the iterative and interlaced generate and test procedure. If in each cycle we only keep B best new attributes (i.e., beam size is B), the procedure takes O(cB^2E) time where c is the pre-determined cycle limit, B is the beam size, and E is the number of examples.

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 consider the relative gain ratio only 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. 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 GainFilter in Figure 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 provided us with the code to LFC so that we could also perform other tests (Thanks to Ricardo Vilalta). Second, we consider the performance of GALA on a number of real world domains. Last, we consider various ablation studies to verify that our system is not overly complicated, i.e. the various components add power to the algorithm.

In all experiments, the parameters of C4.5 and CN2 were set to default values to keep the consistency. For the backpropagation algorithm, we used the best parameter settings after several tries. The learning rate and the momentum was 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)

Artificial Domains

We chose the same four boolean function as did Ragavan & Rendell (1993) and used their training methodology. Each boolean was defined over nine attributes, where four or five attributes were irrelevant. The training set had 32 examples and the remaining 480 were used for testing.

The four boolean functions were:

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

These boolean functions are progressively more difficult to learn, as verified by the experiments with C4.5, or in terms of blurring measure (see Ragavan & Rendell, 1993). 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 1. Each result is averaged over 20 runs. GALA always significantly improved perceptron and backpropagation and usually improved the

Table 1: Accuracy and Hypothesis Complexity Comparison for artificial domains. Significant improvement by GALA over learning algorithms is marked with ∗, and significant difference between GALA+C4.5 (or CN2, perceptron, backprop) and LFC is 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

performance of C4.5 and CN2. 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 outperforms 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 1 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 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. The differences of hypothesis complexities and accuracies are significant at 0.01 level in a paired t-test. 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. In no case was the introduction of new attributes detrimental and in 13 out of 16 cases the generated attribute was useful.

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 from UCI repository several domains that have a mixture of nominal and continuous attributes. These 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 (i.e., included in the final concept description) by the learning algorithm. 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 2 shows the results for 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 in a paired t-test. 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 of 6 for perceptron, 6 out of 7 for backprop) GALA significantly increased the resulting accuracy. Excluding the Diabetes domain GALA always improved the accuracy of both backpropagation and perceptron, mimicking the results for the artificial concepts. For CN2 and C4.5, GALA sometimes improved the performance and never decreased the accuracy.

Ablation Studies

To further demonstrate the value of combining absolute gain ratio with relative gain ratio, we conducted the following studies. First, we evaluated the importance of GainFilter (i.e., combining relative gain ratio with absolute gain ratio). We reran all the experiments, including the artificial and the real domains, but we only used the attributes not passing GainFilter to construct new attributes. The significant decrease of accuracies for all domains indicates that GainFilter could effectively keep promising attributes for constructing new attributes. Those attributes that

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

would not contribute to forming useful new attributes were successfully filtered out by GainFilter. Due to space consideration, we only report the accuracies for C4.5+GALA and C4.5+$GALA^-$ in Table 3, where $GALA^-$ stands for the modified algorithm.

Second, we evaluated the value of relative measure. In any iterative feature construction process, the major difficulty is to determine the usefulness of attributes for later process. One obvious way to avoid overlooking promising attributes is to keep all the attributes, but this is computationally prohibited. Thus beam search naturally comes into play. However, when absolute measure is the only quality criterion, it increases the danger of overlooking promising attributes. For example, a new attribute over its parents with a minor increase of absolute gain ratio by chance may be mistakenly selected in the beam given that the new attribute happens to have high absolute gain ratio. On the other hand, a new attribute with significant increase of gain ratio over its parents, but only with low absolute gain ratio itself may be mistakenly ruled out of the beam though this attribute is promising. Increasing the beam size is one way to solve the problem, but it is difficult to predict the perfect beam size, and arbitrarily increasing the beam size is also computationally prohibited. We first demonstrated the problem mentioned above to emphasize the need of other measure than absolute measure. We reran all the experiments using only the absolute gain ratio with beam size of ten, and compared the results with those of GALA also with the same beam size. GALA's accuracies for heart, wine and promoter domains were significantly better (by 1% to 3%, depending on the domain and the learning algorithm). The results of the other domains were not significantly different. Though the absolute gain ratio beam search sometimes reached the same accuracy as GALA, yet it's search space was much larger (i.e., by 100% to 300% depending on domains). The above results showed that absolute measure does have the problem mentioned earlier. Increasing the beam size could solve the problem, but by how much do we need to increase the beam size? This suggests that we need another measure which combined with absolute measure could not only put the right attributes in a beam and avoid overlooking promising attributes, but also effectively constrain the search space. Thus we introduced the relative measure.

Table 3: Effectiveness Test on GainFilter (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

To further validate the contribution of relative measure, we intentionally removed the relative measure from GainFilter (refer to Figure 4) to increase the search space, and in fact, the new search space covered the old one. Again we reran all the experiments, and found that the accuracies were not significantly different, but search space increased dramatically (i.e., by 25% to 200%, depending on domains). Based on the above studies, we conclude that absolute measure is insufficient, but combined with relative measure could not only avoid overlooking important information, but also effectively constrain the search space.

Conclusion and Future Research

This paper presents a new approach to constructive induction which generates a small number (1 to 3 on average) of new attributes. The GALA method is independent of the learning algorithm so can be used as a preprocessor to various learning algorithm. We demonstrated that it could improve the accuracy of C4.5, CN2, the perceptron algorithm and the backpropagation algorithm. 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.

There is no pruning technique incorporated with the current version of GALA. We suspect that some new attributes generated by GALA may be too complicated to improve the accuracy. Therefore, in one direction of the future research, we will study various pruning techniques. Another limit of the current GALA is that it has only two Boolean operators: "and" and "not". Extending the operators could not only improve the performance of GALA but also expand its applicability.

With hindsight, the results of GALA may be anticipated. Essentially, GALA constructs crisp boolean features. Since some boolean features are outside the space of a perceptron, we should expect that GALA would help this learner the most. In fact, perceptron trainging with GALA generated features was nearly as good as more general symbolic learners. With respect to backpropagation the expectation is similar. While neural nets can learn crisp features, their search bias favors prototypical attributes. Consequently we again expect that GALA will aid neural net learning. While our results were less impressive compared with perceptron and backprop, it is almost more surprising that GALA improves CN2 and C4.5. Both of these algorithms find crisp attributes althought their search biases are somewhat different. However for both algorithms, GALA found useful additional crisp attributes. This demonstrates that GALA's search is different from either of these algorithms. What GALA lacks is the ability to find non-crisp attributes. In our future research we intend to extend GALA to also find non-crisp attributes.

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.

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.

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

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

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.

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

Schlimmer, J. C. "Incremental Adjustment of Representations for Learning" in Proceeding of the Fourth International Workshop on Machine Learning, p79-90, 1987.

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