A Genetic Programming Approach to Constructive Induction

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

ABSTRACT

The performance of inductive learning algorithms greatly depends on their representational biases. Representation determines not only the form and size of the final concept learned, but also the speed of the convergence. One way to mitigate the representation inadequacy is to transform the representation space by generating new features. Unlike most current constructive induction algorithms, in this paper, we propose a stochastic strategy based on genetic programming. It avoids the limitations of most current approaches to feature construction. Also unlike most previous research on constructive induction, our approach is designed as a preprocessing step before standard inductive learning algorithms are applied. We present the results which demonstrate the effectiveness of this new approach on real domains for two different inductive learners : C4.5 and the neural network.

1 Introduction

Poor representation limits the performance of concept learners. One approach to mitigate the limitation is to construct new attributes. The need for useful new attributes has been suggested by many researchers (Matheus, 1989; Ragavan & Rendell, 1993). Constructing new attributes by hand is often difficult. 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), thus yielding improved classification accuracy.

There are currently many different constructive induction algorithms based on the strategy of constructing new attributes, including FRINGE (Pagallo, 1989), CITRE (Matheus & Rendell, 1989), DCFringe (Yang, 1991), LFC (Ragavan & Rendell, 1993), MRP (Perez & Rendell, 1995), etc. According to the construction strategy, current constructive induction algorithms can be categorized into two categories, hypothesis-driven and data-driven. Either strategy has its own advantages and disadvantages.

The hypothesis-driven strategy constructs new attributes based on the hypotheses generated previously. It begins by constructing a hypothesis such as a decision tree, and then examines the hypothesis to build new attributes. After the new attributes are added with the primitive ones, instances are re-described by the new attributes and the primitive ones. The same process is repeated until no good new attributes can be found. Though the hypothesis-driven strategy has the advantages of previous knowledge (i.e., hypotheses), it is highly dependent on the quality of the previous hypotheses. On the other hand, the data-driven strategy constructs and evaluates new attributes by directly analyzing the data. It cannot benefit from previous knowledge and may suffer from computational explosion, but can avoid the strong dependence that impairs the hypothesis-driven strategy.

In this paper, we introduce a new construction strategy, a stochastic approach, which is based on genetic programming. Our new approach avoids not only the strong dependence on previous hypotheses, but also the computational limitation. The new attributes are explicitly represented by Boolean combinations without compromising comprehensibility. Unlike most current constructive induction algorithms, our system is used as a preprocessor to generate new attributes. The new attributes can then be applied to any other standard selective induction algorithm. Thus new attributes have greater applicability and can be evaluated over a wider class of standard inductive learners.

2 GPCI: Genetic Programming for Constructive Induction

2.1 Motivation

One major difficulty with constructive induction is the intractable search space of new attributes. Due to the computational limitation, current constructive induction algorithms either make use of the previous knowledge (i.e., previous hypotheses) or adopt variants of beam search to avoid computational explosion. Unfortunately, for the following reasons, neither approach is effective when the target concept becomes complicated.

First, hypothesis-driven approaches like FRINGE (Pagallo, 1989), CITRE (Matheus & Rendell, 1989), etc., construct new attributes from the hypotheses generated previously by some inductive learner such as a decision tree. They collect the locally repeated patterns in the hypotheses and use these patterns as the basis to create new attributes. Therefore, the quality of the hypotheses determines the usefulness of the new attributes. As the target concept becomes more complicated, the inductive learner they depend on can no longer generate high-quality hypotheses. Consequently, they fail to construct any meaningful new attribute. The strong dependence on previous knowledge plagues these approaches.

Second, in terms of beam search, new attributes are evaluated based on some heuristic measure, but only the new attributes with high heuristic value will be kept in the beam for later process. Though beam search can alleviate the computational limitation, yet it also incurs the danger of overlooking promising new attributes during the construction process. Smaller beam sizes can greatly reduce the computational overhead, but on the other hand, they also rule out some promising new attributes in the early stage of the iterative construction process. These new attributes may become significant in the later construction process. With limited computational resource, constructive induction is mainly a problem of exploring the right portion of the search space of new attributes. Blindly increasing or decreasing the beam size cannot be an effective approach.

For the above reasons, we propose another way to search through the intractable search space, a stochastic approach. With controlled randomness, a stochastic approach has a chance to search the appropriate portion of the search space. One member of the stochastic methods is genetic programming. Based on its stochastic characteristics, genetic programming has a different process bias than other learning algorithms. Provided with an effective fitness function and a sufficient function set (Koza, 1992), genetic programming has the potential to generate useful new attributes through evolution. We introduce GPCI, which is a genetic programming approach to constructive induction.

2.2 Details of GPCI Implementation

The basic idea of GPCI is to use genetic programming to efficiently explore the search space of new attributes; consequently, to construct useful new attributes for any other standard inductive learner.

Most of the current constructive induction algorithms are based on the interleaving model. By interleaving we mean that selective induction and constructive induction are intertwined into a single system. Interleaving model may better tie together selective induction and constructive induction, but on the other hand, constructive induction is also strongly affected by the selective induction process bias, and thus limits its applicability. By keeping these processes separate, the constructive induction algorithm could be used as a preprocessor to any other standard selective induction algorithm. Based on the preprocessing model, we not only increase the applicability of the derived new attributes, but could also test their appropriateness over a wider class of learning algorithms. For all the reasons above, we build GPCI on a preprocessing model.

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 (i.e., threshold).

The first step of GPCI is to transform each real or nominal attributes into two Boolean attributes. 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:

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

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

For each attribute f, find the v in its range such that Pos(f,v) has highest gain ratio (Quinlan, 1993) on the training examples. Then f is transformed into two Boolean attributes, i.e., Pos(f,v) and Neg(f,v), and are put into Bool. After the transformation, all the Boolean attributes will be used as the building blocks to construct new attributes.

```
Given: Primitive attributes P, training examples E,
       threshold T, reproduction probability Pr,
       crossover probability Pc,
       max number of generations G,
       and new attributes NEW
(Note: NEW is empty when GPCI invoked the first time)
Return: a set of new attributes NEW

Procedure GPCI(P,E,T,Pr,Pc,G,NEW)
  If (size(E) > T)  and  (E is not all of same class)
    Then Booleanize real or nominal attributes into Bool.
```

```
Initialize Population with Bool and NEW
Pool = GP(Population,Pr,Pc,G)
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 = GPCI(P,Ei,T,Pr,Pc,G,NEW)
  N = union of N and NEWi
  NEW = union of NEW and N
  Return NEW
 end For
Else Return empty set
```

Figure 1 Pseudo-code of GPCI

The second step of GPCI, procedure GP, is to apply genetic programming to select the candidate attributes for attribute construction and then evolve to the next generation of new attributes. The search space of genetic programming is defined by terminal set and the function set (Koza, 1992). In procedure GP, the terminal set consists of the primitive attributes that are used to represent the instances; the function set contains only two Boolean operators, AND and NOT, in the current version of GPCI.

Each individual in a GPCI population is an attribute that can be either a primitive attribute or a derived new attribute. New population is generated by applying reproduction operator and crossover operator. Mutation operator is not used in the current version of GPCI. The fitness of a new individual (i.e., a new attribute) is evaluated by a heuristic function that combines a absolute measure and a relative measure. The absolute measure is used to evaluate the quality of a new attribute over the training instances without regard to how it is constructed. Examples of such measures include entropy, information gain, gain ratio, error rate, etc. While it is important that a new attribute performs well on the training instances, it is also important that it has significant improvement over its parents. We refer to this as a relative measure since it depends on the quality improvement relative to a new attribute over its parents. For GPCI, we select gain ratio (Quinlan, 1993) as an absolute measure and derive the relative measure as the following.

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:

$$RGR_{UPPER}(A) = max\{\frac{GR(A)-GR(A1)}{GR(A)}, \frac{GR(A)-GR(A2)}{GR(A)}\}$$

$$RGR_{LOWER}(A) = min\{\frac{GR(A)-GR(A1)}{GR(A)}, \frac{GR(A)-GR(A2)}{GR(A)}\}$$

To combine the absolute measure and the relative measure, we define the fitness of attribute A as:

$Fitness(A) = W1 * GR(A) + W2 * RGR_{UPPER}(A) + W3 * RGR_{LOWER}(A)$ where W1, W2 and W3 are user-defined weights.

The fitness function exploits the synergy of the conjunction over the value of the individual attributes. Table 1 shows the Koza tableau.

2.3 Related Works

As mentioned earlier, there are many different constructive induction algorithms. In this section, due to the space limitation, we only discuss recent related works also based on genetic algorithms or genetic programming, for example, COEV (Abramson & Hunter, 1996), GP-Knn (Raymer *et. al.*, 1996), GA-ID3 (Bala *et. al.*, 1995).

COEV (Co-evolution) interleaves multiple learning algorithms with a search through the space of possible representations. The co-evolution arises through the evolution of the parameters of the learning algorithms and the evolution of their data representations. Each individual in a COEV population has both a vector of parameters that control the inductive learner and a set of feature definitions. In each generation, representations are extracted from the output of each inductive learner, and then compete for evolving to the next generation. This co-evolution learning is basically a hypothesis-driven approach since new representations are derived from the hypotheses (i.e., the output) generated by the inductive learners. As mentioned earlier, one major drawback of hypothesis-driven approaches is that the performance is strongly dependent on the quality of the hypotheses (i.e., the output) generated by the inductive learner. Secondly, COEV intertwines multiple learners in the evolution process. Its efficiency can be seriously impaired by a slow learner such as the neural network.

GP-Knn is designed to address the feature extraction problem with two results. First, it finds a list of important features for distinguishing the data. Second, for the list of features, it finds the relation and the associated weights. Unfortunately, like COEV, GP-Knn also depends on the performance of another inductive learner, i.e., the nearest-neighbor algorithm. The sensitivity of the nearest-neighbor algorithm to irrelevant features may decrease GP-Knn's effectiveness.

Unlike the above systems, GA-ID3 combines genetic algorithms and decision tree learning (i.e., ID3) to evolve optimal subset of discriminatory features for pattern classification. For a given feature subset, ID3 (Quinlan, 1986) is invoked to produce a decision tree. The classification performance of the decision tree is used as a measure of fitness for the feature subset, which, in turn, is used by the GA to evolve better feature set. Conse-

Table 1 Tableau for GPCI

Objective	Construct useful new attributes
Terminal Set	primitive attributes
Function Set	AND, NOT
Fitness	see definition
Parameters	population size = 500 max number of generations = 10 crossover prob = 80% reproduction prob = 60% threshold T = 10 W1 = 0.50 W2 = W3 = 0.25

quently, GA-ID3 is plagued by the same undesired strong dependence as we see in the above systems.

Our new approach, GPCI, is different from all the above systems. It is independent of any inductive learner. Its attribute construction process does not depend on the hypotheses/outputs of any inductive learner. Therefore, it can avoid the strong dependence and its efficiency will not be compromised by any inductive learner. In addition, based on a preprocessing model, GPCI could be applied to any other standard inductive learner as an attribute construction preprocessor; thus, it has a wider applicability than most current constructive induction algorithms.

3 Experimental Results of Real Domains

There are two objectives of the experiment. Firstly, we want to test whether GPCI can improve the performance of other standard inductive learners by adding new attributes. Secondly, we want to justify that our new approach is competitive compared with current constructive induction algorithms. We used C4.5 (Quinlan, 1993), which is a representative decision tree algorithm, and the neural net with backprop as the standard inductive learners in our experiment. And we used two constructive induction algorithms, LFC (Ragavan & Rendell, 1993) and GALA (Hu & Kibler, 1996), for comparison, where LFC is based on the interleaving model and GALA is based on the preprocessing model. We tested our new approach on twelve real domains in UCI machine learning databases. The test domains are summarized in table 2.

In each case, two thirds of the examples form the training set and the remaining examples form the test set. The results of predictive accuracy on testing sets are shown in table 3. Each result is averaged over 30 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 multi-class domains. The fifth and six columns present the new accuracies of the neural net and C4.5 after adding the new attributes generated by GALA. The new accuracies of the neural net and C4.5 with the addition of the new attributes generated by GPCI are shown in column seven and eight. Significant improvement (0.05 level in one tailed paired t-test) by GPCI over the inductive learner is marked with "1", and significant difference of accuracy between GPCI and LFC (or GALA) is marked with "2" (or 3).

The experimental results showed that GPCI significantly improves the inductive learners on most of the domains, and is very competitive compared with other constructive induction algorithms. In no case was the introduction of the generated attributes harmful to the learning algorithm.

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

4 Future Work and Conclusion

We plan to explore GPCI in several ways. In particular, we want to extend the function set in GPCI. The function set determines not only the search space but also the expressiveness of new attributes. By extending the function set, we could have a more flexible representation of new attributes. Secondly, we want to carry a systematic analysis for empirical characterization and evaluation of GPCI. In this study, we systematically generate the artificial domains and concepts to determine the effectiveness of GPCI. Finally, we want to extend the capabilities of utilizing various forms of background

Table 2 Domain Summary

Domain	No. of Examples	No. of Real Attr/No. of Nominal Attr	No. of Classes
Breast Cancer	699	9/0	2
Heart Disease	303	5/8	2
Liver Disorder	345	7/0	2
Ionosphere	351	34/0	2
Lymphography	148	3/15	4
Promoter	106	0/57	2
Horse Colic	300	7/15	2
Hayes Roth	160	0/4	3
Glass	214	9/0	7
Iris	150	4/0	3
Solar Flare	323	9/3	6
Pima Diabetes	768	8/0	2

Table 3 Results of Real Domains

Domain	NN	C4.5	LFC	NN+GALA	C4.5+GALA	NN+GPCI	C4.5+GPCI
Breast Cancer	92.7	94.5	94.2	95.1	95.8	94.8^{1}	95.5^{12}
Heart Disease	74.3	72.3	75.2	76.9	76.7	76.9^{12}	76.6^{12}
Liver Disorder	66.1	62.1	62.4	68.7	65.4	68.1^{12}	64.7^{12}
Ionosphere	87.3	90.1	90.3	90.3	92.2	90.6^{1}	92.5^{12}
Lymphography	87.1	76.0	N/A	87.3	77.8	87.3	79.5^{13}
Promoter	71.9	73.9	75.1	77.1	79.5	78.2^{12}	80.9^{12}
Horse Colic	81.2	82.3	78.5	81.7	81.0	82.7^{2}	83.0^{23}
Hayes Roth	70.2	74.9	N/A	80.9	82.9	80.9^{1}	82.9^{1}
Glass	60.9	63.8	N/A	61.8	62.6	62.1	63.1
Iris	95.6	94.1	N/A	95.6	94.1	95.6	94.1
Solar Flare	64.9	69.5	N/A	65.9	70.2	65.9	70.2
Pima Diabetes	67.1	71.6	71.1	67.9	72.0	67.9	71.8

knowledge. Background knowledge could help constrain the search space and provide guidance of selecting appropriate construction operators and attribute candidates.

Representation inadequacy has long been noticed as a major problem for typical inductive learners, and constructive induction is one solution to this problem. This paper presents a new stochastic approach which is not limited to the low quality of the initial hypotheses produced by a greedy inductive learner or to insufficient lookahead search. Instead, GPCI based on genetic programming selects candidate attributes to evolve new attributes. The contribution of GPCI could be summarized as below. First, GPCI applies a stochastic approach to search through the intractable search space of new attributes. Through evolution, this stochastic search strategy could effectively search the right portion of the search space of new attributes without the limitations of the strong dependence on the hypotheses produced by some inductive learner or the inadequacy of some limited lookahead beam search. Second, GPCI is independent of any learning algorithm. It is basically an off-line preprocessor that could be applied to any standard inductive learner. The goal of GPCI is to generate useful new attributes for other learning algorithms. Using this approach, we demonstrated significant improvement in several real domains and no degradation in accuracy in any domain. The usefulness of the generated attributes was also demonstrated for two different learning algorithms, C4.5 and the neural network.

5 Acknowledgement

I thank the paper reviewers for their valuable comments and suggestions.

References

Abramson, M. & Hunter, L."Classification Using Cultural Co-evolution Genetic Programming", in Proceeding of Genetic Programming, p249-254, 1996.

Bala, J., Huang, J., Vafaie, H., Dejong, K., Wechsler, H."Hybrid Learning Using Genetic Algorithms and Decision Trees for Pattern Classification", in Proceeding of International Joint Conference on Artificial Intelligence, p719-724, 1995.

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.

Hu, Y. & Kibler, D. "Generation of Attributes for Learning Algorithms", in Proceeding of the 13th National Conference on Artificial Intelligence, p806-811, 1996.

Koza, J. "Genetic Programming: On the Programming of Computers by Means of Natural Selection", MIT Press, 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.

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

Pagallo, G. "Learning DNF by Decision Trees", in Proceeding of the 11th International Joint Conference on Artificial Intelligence, 1989.

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. "Inductive Decision Trees", Machine Learning, 1986.

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

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

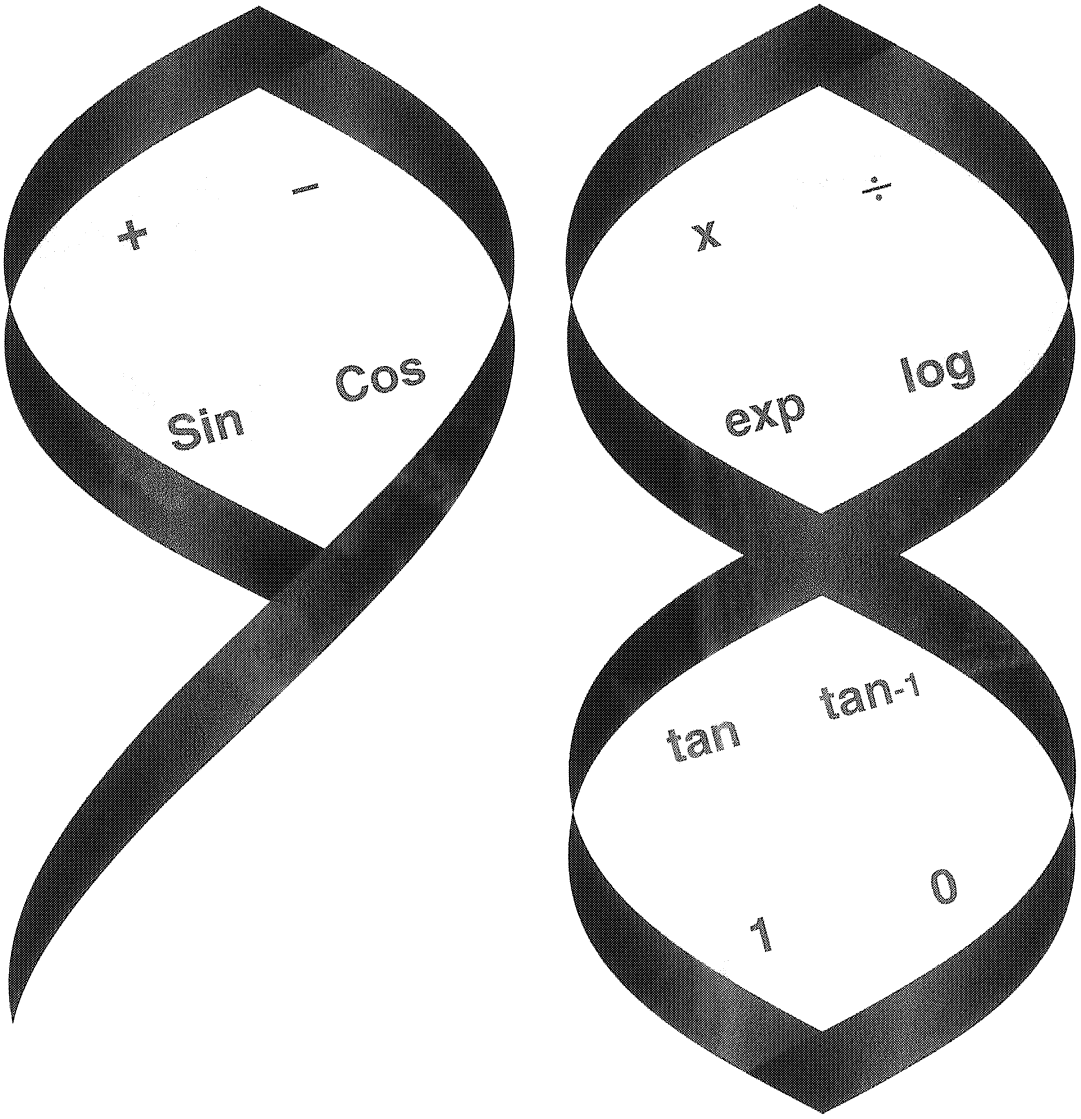
Raymer, M., Punch, W., Goodman, E., Kuhn, L."Genetic Programming for Improved Data Mining – Application to the Biochemistry of Protein Interactions", in Proceeding of Genetic Programming Conference, p375-380, 1996.

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.

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.

CONFERENCE PROCEEDINGS

Genetic Programming



edited by
John R. Koza
Wolfgang Banzhaf
Kumar Chellapilla
Kalyanmoy Deb
Marco Dorigo
David B. Fogel
Max H. Garzon
David E. Goldberg
Hitoshi Iba
Rick L. Riolo

Proceedings of the Third Annual Genetic Programming Conference

July 22–25, 1998
University of Wisconsin, Madison, Wisconsin