ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

Application of Genetic Programming for Pattern Recognition

Mansoor Farooq

School of Science & Technology, Department of Computer Science, Shri Venkatashwara University, U.P, India

Abstract: Genetic programming combines and extends discrete decision theory with the principles of genetic and natural selection. The programs may be in the form of decision trees or diagram. The decision trees and diagrams are used in many discipline, genetic programming has many applications. Among those applications is pattern recognition. Different genetic programming techniques exist. This section describes a general technique for programs that use mathematical function. A function is routine that take one or two arguments, performs some function and returns value. The arguments with the routine are also functional routines, the resulting programs is like a tree in which each node represents a functional routine and each subtree an argument. Genetic programming with subtree crossover technique will be used that evolves a population over much iteration until some termination is satisfied. During each iteration the existing population is replaced by a new population that is derived from the existing population. The primary operations of reproduction and crossover are used for all problems. The two operations were sufficient for most of the problem to which the technique was applied in [9]

Keywords: Genetic Programming, Crossover, Decision Tree, Decision Diagram and Pattern Recognition.

I. INTRODUCTION

Genetic programming is a branch of genetic algorithms. Genetic programming creates computer programs in the lisp or scheme computer languages as the solution. Genetic algorithms create a string of numbers that represent the solution. The best computer program that appeared in any generation, the best-so-far solution, is designated as the result of genetic programming [1] Genetic programming is useful in finding solutions where the variables are constantly changing. GP evolves computer programs, traditionally represented in memory as tree structures. [2] Trees can be easily evaluated in a recursive manner. Every tree node has an operator function and every terminal node has an operand, making mathematical expressions easy to evolve and evaluate. Thus, traditionally GP favours the use of programming languages that naturally embody tree structures for example Lisp.

Genetic programming combines and extends discrete decision theory with the principles of genetic and natural selection. The programs may be in the form of decision trees or diagram. The decision trees and diagrams are used in many discipline, genetic programming has many applications. Among those applications is pattern recognition. The use of the word "pattern" in this research differs from that found in some literature on pattern recognition. In that literature, a pattern is a representation of a scene with the things to be recognized. In contrast, in this research patterns must be discoveries. Each pattern is as cluster in the recognition problem space. It consists of the features ranges to which some a representation is compared for classification. In other words, the pattern is a set of criteria by which some representations of the things to be recognized can be distinguished from representation of other similar and not-so-similar things. Those criteria identify properties that are common on usually multiple representations of the things to be recognized. Hence, the use of the word "pattern".

II. THE DECISION TREE OR DIAGRAMS

Since the 1950's, many concept learning systems have been implemented as decision-tree construction algorithm [3][4][5] to construct a tree from training data is easy.

The Decision Tree

A decision tree is a set of test sequences by which one can reach a conclusion. Each of the tree's branching nodes represents a test for which the outcome determines the path taken to be the next node. Eventually it will be a leaf node representing some conclusion. In a decision tree used for pattern recognition, that conclusion is the class of a representation. Thus the tests concern the pattern characteristics by which the representation belongings to any class can be distinguished among representation belongings to other classes.

Decision Diagrams

The decision tree is folded back on top of itself to create a decision diagram. The diagram differs from tree in that some of the diagram's node may have more than one parent. This allows the diagram to encode the same information in a more compact form. Although the compactness of the diagram is important for some applications, the important point here is that two different forms are equivalent and each can be easily converted into the other.

Decision Location

There are two approaches. When using a decision tree, many authors believe that one makes a series of decisions to reach a conclusion, for these the outcome of a test in a branching node is a decision. Other authors consider the

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

conclusion in a leaf node to be the decision. One might traverse one decision tree in order to reach a conclusion that answer a question in another decision tree.

Feature Selection and Extraction during Classification
Features are selected for extraction while using a decision tree for classification. The test at each branching node extracts some feature. Then based on the value of that feature, the decision tree either selects the nest feature to be extracted or arrives at a conclusion. That is why decision trees are frequently used for diagnostic setting. This method allows classifiers to avoid extracting useless features for unnecessary tests. The important point is that a decision tree identifies the context in which specific features should be selected for extraction.

Decision Equivalence
Two decision trees are decision equivalent if their conclusions are in agreement for every possible test sequence. Three identities apply to decision equivalent, but not structurally equivalent [6] [7] [8]. Suppose that N branches are attached to a decision tree at some branching node. Then, one of those branches can be selected by answering some question Q. Since each branch leads to a subtree there are N possible subtrees, S_1 through Sn.
$Q.\ (S_1, S_2, ...S_n)$
Using this notation, the three identities are:

Idempotence:
$Q.\ (X, ..., X) = X$
Repetition:
$Q.(X_1,.....,X_r\text{-}.....Q(Y_1,...,Y_r\text{-}.....Y_r,.....,Y_r+.....,Y_n),\ X_r+,....X_n) = Q.(X_1,....,X_r\text{-},Y_r,\ X_r+,.....,X_n)$

Transposition:
$Q_1.(Q_2.(X_{11},.....X_{1m}),.....,Q_2.(X_{n1},....X_{nm}))=$
$Q_2.(Q_1.(X_{11},...X_{n1}),.....Q_1.(X_{1m},.....,X_{nm}))$

Checking Decision Equivalence
The decision equivalence of two trees can be checked in different ways. One way is to compare the conclusion of both trees for all possible instances. This is impractical. A better ways is based on discrete decision theory. [6][7][8] A tree is simply reduced if it cannot be made any smaller by repeatedly applying the repetition identity and then repeatedly applying the idempotence identity.

Decision Tree Construction
Two conventional decision-tree construction algorithms are well know. One is the "classification and regression tree" (CART) algorithm [16] and the other is C4.5 algorithm [10]. Both the algorithm consists of two steps. The first step grows a tree that correctly or almost correctly classifies every training datum. The resulting tree may "overfit" that data. This means that the tree obtains its accuracy on those training data in a way that degrades its accuracy on other instances of the recognition problem. The second step adjusts for that possibility by pruning the tree. [18] compares the performance of the following algorithm:

- C4.5 variants

C1 – C4.5 with its –m1 option. A leaf is split when it classifies 1 or more examplars of the second most-frequent class to visit the leaf.
C2 – C4.5 with its default setting. A leaf is split when it classifies 2 or more examplars of the second most-frequent class to visit the leaf.

- Incremental tree-inducer (ITI) variants

I1 – ITI set to split a leaf when the leaf classifies 1 or more examplars of the second most-frequent class to visit the leaf.
I2 – ITI set to split a leaf when the leaf classifies 2 or more examplars of the second most-frequent class to visit the leaf.
IE – Like I1 but runs in error-correction mode instead of batch mode.

- Direct-metric tree-inducer (DMTI) variants

DE – DMTI with the expected quantity of tests for classification as the direct metric.
DL – DMTI with the quantitiy of leaves as the direct metric.
DM – DMTI with the minimum description length as the direct metric.

III. GENETIC PROGRAMMING

Genetic programming combines and extends discrete decision theory with the principles of genetic and natural selection. The programs may be in the form of decision trees or diagram. The decision trees and diagrams are used in many discipline, genetic programming has many applications. Among those applications is pattern recognition. Different genetic programming techniques exist. This section describes a general technique for programs that use mathematical function.

A function is routine that take one or two arguments, performs some function and returns value. The arguments with the routine are also functional routines, the resulting programs is like a tree in which each node represents a functional routine and each subtree an argument. It means that subtree crossover can be used in a variation of genetic algorithm to evolve a population of increasingly fit functional program.[9][1]

Genetic programming with subtree crossover is a technique that evolves a population over much iteration until some termination is satisfied. During each iteration the existing population is replaced by a new population that is derived from the existing population. The primary operations of reproduction and crossover are used for all problems. The two operations were sufficient for most of the problem to which the technique was applied in [9]

Reproduction
An operation that creates a replica of a program in the existing population, the existing program is randomly selected with a preference for those individuals that are more fit.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

Crossover
An operation that creates two new programs by wrapping subtrees between what are otherwise replica of two programs in the existing population. These existing programs are randomly selected with a preference for programs that are fit.

Within each of the program, a random node is chosen as the crossover point. The subtree rooted at that point is copied into the replica of the other tree at that tree's crossover point and vice versa.

The two modified replica became part of the new population. The secondary operations are of little benefit and thus rarely used in [9] the secondary operations are:

Mutation
An operation that creates a new program by inserting a randomly-generated subtree at a random point in what would otherwise be a replica of an existing program.

Permutation
An operation that creates a new program by inserting a randomly swapping the branches that emanate from a random branching node within a replica of an existing program.

IV. EXPERIMENT

To achieve comparable accuracy on the training data, the genetic programming technique required both too much memory and too much computation [10]. The hybrid needed much less memory, but still too much computation [10] [11]. Two set of experiments were conducted in which genetic programming with subtree crossover was used in an attempt to evolve small, accurate decision trees from large set of training data.

The first set of experiments compared the performance of this the genetic algorithm technique to that of C4.5 software.[10] The result of the experiment is explained, one was lack of mutation operators and the other inappropriate crossover operator. Without mutation, new branching node tests could not be introduced into the population as it evolved. Instead all the needed tests had to exist in the initial population.

This meant that the population had to be quit large. In the second, experiment, this problem was reduced by using C4.5 to partially address the problems introduced by subtree crossover. The use of C4.5 introduced new branching node tests into the population as it evolved. Thus the combination of subtree crossover with C4.5 acted like a mutation operator.

The problem is that subtree crossover does not preserve decision tree accuracy even when the parents are identical.

For example in Figure (a), Figure (b) and Figure (c) the two identical parents on the left accurately as shown in the figure classify every instance of the recognition problem.

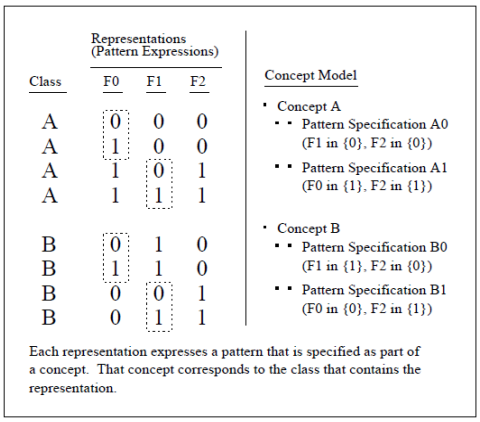

Fig. (a) Pattern Expression & Specification

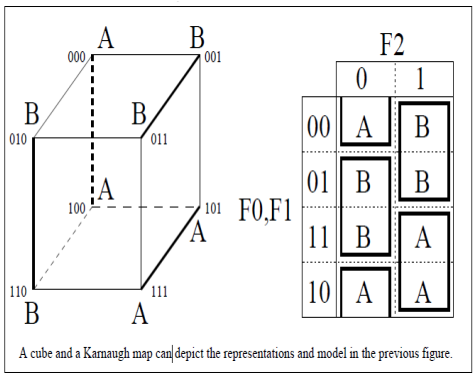

Fig. (b) A Problem Space

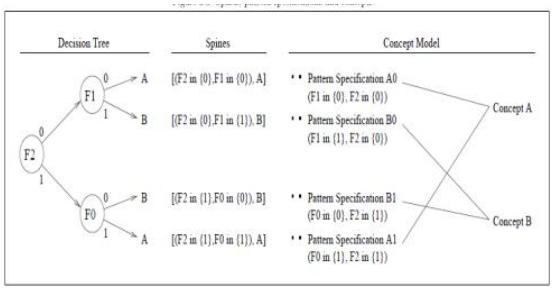

Fig. (c) Pattern Specifications and Concept

The problem requires a decision tree with at least four leaf nodes, one for each pattern cluster. But each of the children divides the problem space into two clusters. After reduction by the Transposition and Idempotence identities, each child has two leaf nodes. As a result accuracy drops, because the information (subtree) extracted from each parent was out of context after its insertion at the crossover point in the replica of the parent. In Figure (d) the randomly chosen crossover point in one parent is at the node that tests feature F0, and in the other parent at the node that tests feature F1.

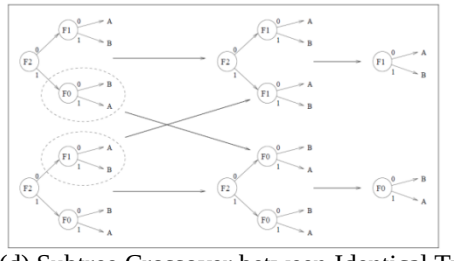
Fig. (d) Subtree Crossover between Identical Trees

DOI 10.17148/IJARCCE.2015.4403

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

The context of the crossover point in each parent is defined by the test-outcome pairs on the path from the parent's root to, but not including, the crossover point. Since each parent accurately classifies every instance of the recognition problem. However, genetic programming with subtree crossover does not automatically do so. It extracts a randomly chosen subtree to replace the existing subtree. When subtree crossover is used in genetic programming, it can be applied to any tree of functions. A decision tree is just one such type tree. Subtree crossover crosses two randomly selected subtrees without any regards for their context. The sub problems can be easily compared when genetic algorithm is evolving decision trees. The method appears to depend on how meaning is encoded within the tree and that encoding depends on the nature of the problem. As a result, some new operators have been conceived for subtree crossover [12][13][14][15]. The new operators try to cause less damage by restricting the subtrees eligible for crossover. The operators first align two trees and then cross over only subtrees that exist in similar structural contexts. For example, crossover between two identical parents can be constrained to produce an identical child.

V. CONCLUSION

By creating a genetic algorithm specifically for the decision programs includes both (trees and diagrams) this research has contributed to study of pattern recognition, machine learning and evolutionary computation. The primary contribution of this research springs from the realization that the information encoded in decision tree can be easily recombined in such a way that its meaning is preserved. The technique is a genetic algorithm with crossover and mutation operators that are specifically designed for decision programs (tree and diagrams). The algorithm extends discrete decision theory in the search for those decision programs that best satisfy some user defined criteria. In this research those criteria concerned how a computer might perform a machine learning task to achieve pattern recognition.

VI. REFERENCES

[1]. John R. Koza. "Non-linear genetic algorithms for solving problems", 1990, patent 4,935,877, United States Patent and Trademark Office, Alexandria, Virginia, USA.
[2]. Cramer "PROCEEDINGS OF AN INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS" Carnegie-Mellon University Pittsburgh, PA APPLICATIONS.
[3]. Earl B. Hunt and Carl I. Hovland, "Programming a model of human concept formation", [WJCC – 1961, pages 145 – 155].
[4]. Earl B. Hunt, "Concept Learning: An Information Processing Problem", 1962, John Willey and Sons, New York, New York, USA.
[5]. Earl B. Hunt, Janet Martin and Philip J. Stone, "Experiments in Induction", 1966, Academic Press, New York, New York, USA.
[6]. J. R. B. Cockett, "Decision Expression Optimization", 1987, Fundamentals Informaticae, Volume X, pages 93 – 114.
[7]. J. R. B. Cockett, "Discrete decision theory: Manipulations", 1987, Theoretical Computer Science, Volume 54, pages 215 – 236.
[8]. J. R. B. Cockett and J. A. Herrera, " Decision Tree Reduction", 1990, Journal of the Association for Computing Machinery, Volume 37, Pages 815 – 842.
[9]. john R. Koza, " Non-linear genetic algorithms for solving problems", 1990, patent 4,935,877, United States Patent and Trademark Office, Alexandria, Virginia, USA.
[10]. J. Ross Quinlan, "C4.5, Programs for Machine Learning, 1993, Morgan Kaufmann Publishers, San Mateo, California, USA, ISBN: 1-55860-2380-0.
[11]. M. D. Ryan and V. J. Rayward-Smith, "The Evolution of Decision Trees", 1998, [Koza et al. – 1998], pages 350 – 358.
[12]. Patrik D'haeseleer, "Context Preserving Crossover in Genetic Programming", 1994, [ICEC – 1994], volume 1, pages 256 – 261, DOI: 10.1109/ICEC.1994.350006.
[13]. Peter Nordin, Wolgang Banzhaf and Frank D. Francone, "Efficient Evolution of Machine Code for CISC Architectures Using Instruction Blocks and Homologous Crossover", 1990, [Spector, Langdon, O'Reilly and Angeline – 1999], chapter 12, pages 275 – 299.
[14]. Riccardo Poli and W. B. Langdon, "On the Search Properties of Different Crossover Operators in Genetic Programming", [Koza et al. – 1998], pages 293 – 301.
[15]. W. B. Langdon, "Size Fair and Homologous Tree Crossovers for Tree Genetic Programming", Genetic Programming and Evolvable Machines, volume 1, pages 95 – 119.
[16]. Leo Breiman, Jerome H. Friedman, Richard A. Olshen and Charles J. Stone, "Classification and Regression Trees", 1984, Wadsworth International Group, Belmont, California, USA, ISBN: 0-534-98053-8 (hardcover) and 0.534-98054-6 (paperback).
[17]. Paul E. Utgoff, Neil C. Berkman and Jeffery A. Clouse, " Decision Tree Induction Based on Efficient Tree Restructturing", 1997, Machine Learning, Volume 29, pages 5 – 44.

DOI 10.17148/IJARCCE.2015.4403