ISSN No. 0976-5697

Volume 4, No. 3, March 2013 (Special Issue)

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Generate classifier for Genetic Programming of Multicategory Pattern Classification Using Multiclass Microarray Datasets

Anwar Mohd Mansuri, Scholar
Department of Information Technology, MIT
Ujjain, India
anwar.iter@gmail.com

Deepali Kelkar, Reader
Department of Information Technology, MIT
Ujjain, India
bdeepali_b@yahoo.co.in

***Abstract*:** In this paper a multiclass classification problem solving technique based on genetic programming is presented. This paper explores the feasibility of applying genetic programming (GP) to multicategory pattern classification.GP can discover relationships among observed data and express them mathematically Feature selection approaches have been widely applied to deal with the small sample size problem in the analysis of microarray datasets. Multiclass problem, the proposed methods are based on the idea of selecting a gene subset to distinguish all classes. However, it will be more effective to solve a multiclass problem by splitting it into a set of two- class problems and solving each problem with a respective classification system, Data mining deals with the problem of discovering novel and interesting knowledge from large amount of data. The results obtained show that by applying Modified crossover together with Point Mutation improves the performance of the classifier. A comparison with the results achieved by other techniques on a classical benchmark set is carried out.

Keywords: Microarray; Classifier; Genetic Programming; Mutation, Crossover

I. INTRODAUCTION

Data classification represents perhaps the most commonly applied supervised data mining technique. It consists in generating from a set of class-labeled training examples a set of grouping rules which can be used to classify future patterns. There are many methods used to face the classification task. They include decision-tree methods which operate performing a successive partitioning of cases until all subsets belong to a single class. The classification problem becomes very hard when the number of possible different combinations of parameters is high.Hense we used Different Operators for solving different operators for solve the problem. Genetic programming (GP) is an automated method for creating a working computer program from a high-level problem statement of a problem. Genetic programming starts from a high-level statement of "what needs to be done" and automatically creates a computer program to solve the problem.

II. GENETIC PROGRAMMING

In artificial intelligence genetic programming (GP) is an evaluation algorithm based methodology inspired by biological evolution to find computer programs that perform a user-defined task. It is a specialization of Genetic Algorithm (GA) where each individual is a computer program. It is a machine learning technique used to optimize a population of computer programs according to a fitness landscape determined by a program's ability to perform a given computational task. One of the central challenges of computer science is to get a computer to do what needs to be done, without telling it how to do it. Genetic programming addresses this challenge by providing a method for automatically creating a working computer program from a high-level problem statement of the problem. Genetic programming achieves this goal of automatic programming (also sometimes called program synthesis or program induction) by genetically breeding a population of computer programs using the principles of Darwinian natural selection and biologically inspired operations. The operations include reproduction, crossover mutation, and architecture-altering operations patterned after gene duplication and gene deletion in nature. Genetic programming is a domain-independent method

That genetically breeds a population of computer programs to solve a problem. Specifically, genetic programming iteratively transforms a population of computer programs into a new generation of programs by applying analogs of naturally occurring genetic operations. The genetic operations include crossover, mutation, reproduction, gene duplication, and gene deletion.

III. PROPOSED WORK

In this paper we have designed a Multiclass Classifier using different Operations.

A. How Genetic Programming Work:

Genetic programming starts with a primordial ooze of thousands of randomly created computer programs. This population of programs is progressively evolved over a series of generations. The evolutionary search uses the Darwinian principle of natural selection (survival of the fittest) and analogs of various naturally occurring operations, including crossover, mutation, gene duplication, gene deletion. Genetic programming sometimes also employs developmental processes by which an embryo grows into fully developed organism.

B. Executional Steps of Genetic Programming:

Genetic programming typically starts with a population of randomly generated computer programs composed of

CONFERENCE PAPER
II International Conference on
"Advance Computing and Creating Entrepreneurs (ACCE2013)"
On 19-20 Feb 2013
Organized by
2nd SIG-WNs, Div IV & Udaipur Chapter , CSI , IEEE Computer Society Chapter India Council ,
IEEE Student Chapter Geetanjali Institute of Technical Studies, Udaipur, Rajasthan, India

the available programmatic ingredients. Genetic programming iteratively transforms a population of computer programs into a new generation of the population by applying analogs of naturally occurring genetic operations. These operations are applied to individual(s) selected from the population. The individuals are probabilistically selected to participate in the genetic operations based on their fitness (as measured by the fitness measure provided by the human user in the third preparatory step). The iterative transformation of the population is executed inside the main generational loop of the run of genetic programming. The exceptional steps of genetic programming (that is, the flowchart of genetic programming) are as follows:

a. Randomly create an initial population (generation 0) of individual computer programs composed of the available functions and terminals.
b. Iteratively perform the following sub-steps (called a generation) on the population until the termination criterion is satisfied:
c. Execute each program in the population and ascertain its fitness (explicitly or implicitly) using the problem's fitness measure.
d. Select one or two individual program from the population with a probability based on fitness (with reselection allowed) to participate in the genetic operations.
e. Create new individual program for the population by applying the following genetic operations with specified probabilities:
f. Reproduction: Copy the selected individual program to the new population.
g. Crossover: Create new offspring program for the new population by recombining randomly chosen parts from two selected programs.
h. Mutation: Create one new offspring program for the new population by randomly mutating a randomly chosen part of one selected program.
i. Architecture-altering operations: Choose an architecture-altering operation from the available repertoire of such operations and create one new offspring program for the new population by applying the chosen architecture-altering operation to one selected program.
j. After the termination criterion is satisfied, the single best program in the population produced during the run (the best-so-far individual) is harvested and designated as the result of the run. If the run is successful, the result may be a solution (or approximate solution) to the problem.

IV. FLOWCHART OF GENETIC PROGRAMMING

The figure below is a flowchart showing the exceptional steps of a run of genetic programming. The flowchart shows the genetic operations of crossover, reproduction, and mutation as well as the architecture-altering operations. This flowchart shows a two-offspring version of the crossover operation. Genetic programming starts with an initial population of computer programs composed of functions and terminals appropriate to the problem. The individual programs in the initial population are typically generated by recursively generating a rooted point-labeled program tree composed of random choices of the primitive functions and terminals.

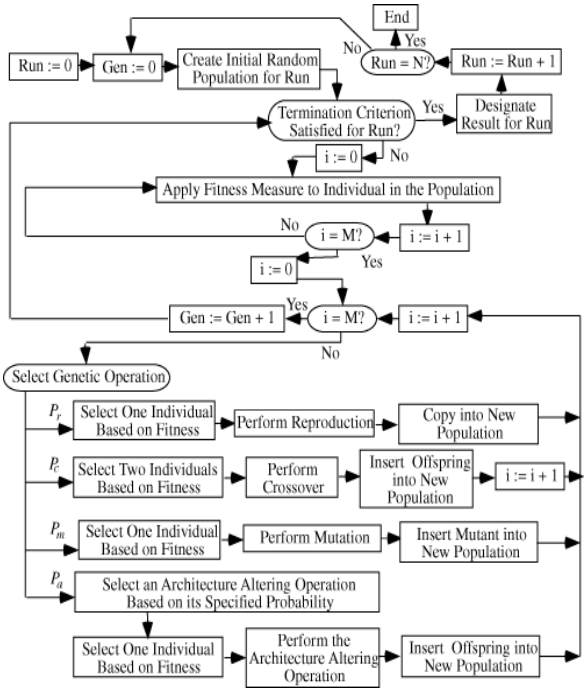

Figure.1.Flow Chart of Genetic Programming

A. Mutation Operation:

In the mutation operation, a single parental program is probabilistically selected from the population based on fitness. A mutation point is randomly chosen, the sub tree rooted at that point is deleted, and a new sub tree is grown there using the same random growth process that was used to generate the initial population. This mutation operation is typically performed sparingly (with a low probability of, say, 1% during each generation of the run).

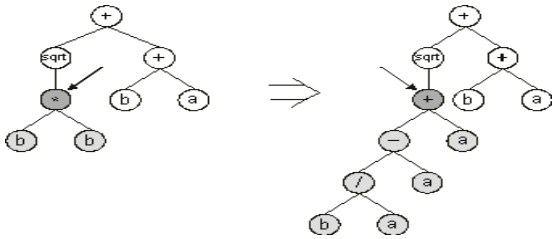

Figure.2.Tree mutation in genetic programming.

We used Point Mutation technique for mutation, Point Mutation is a special technique in which we randomly selected an individual from the population and from this randomly selected individual we generate the two children. From this two generated children one is rejected on the basis of fitness the children with the lower fitness is rejected now we compare the selected individual with the parent and compare its fitness with the parent if the fitness of parent is better than the child than with a probability of 0.5 parent is transferred to the next generation otherwise children is transferred to the next generation. By applying this special mutation technique we are sure that the generated individual does

CONFERENCE PAPER
II International Conference on
"Advance Computing and Creating Entrepreneurs (ACCE2013)"
On 19-20 Feb 2013
Organized by
2nd SIG-WNs, Div IV & Udaipur Chapter , CSI , IEEE Computer Society Chapter India Council ,
IEEE Student Chapter Geetanjali Institute of Technical Studies, Udaipur, Rajasthan, India

not reduce the fitness and also provide the diversity among the individuals

B. Crossover Operation:

In the crossover, operation, two parental programs are probabilistically selected from the population based on fitness. The two parents participating in crossover are usually of different sizes and shapes. A crossover point is randomly chosen in the first parent and a crossover point is randomly chosen in the second parent. Then the sub tree rooted at the crossover point of the first, or receiving, parent is deleted and replaced by the sub tree from the second, or contributing, parent. Now apply the elitism on the generated children and compare the fitness of the children with the parent if the fitness of the parent is greater than the children than with the probability of 0.25 parent is transferred to the next generation otherwise children is transferred to the next generation.

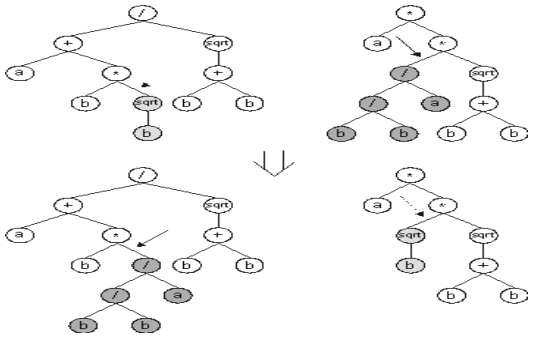

Figure.3.Tree crossovers in genetic programming.

C. Reproduction Operation:

The reproduction operation copies a single individual, probabilistically selected based on fitness, into the next generation of the population..

a. Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses).

V. CO EVOLUTIONARY GENETIC ALGORITHM

Step 1. Begin
Step 2. Define pop-size as desired population size
Step 3. Randomly initialize pop-size population
Step 4. While (Ideal best found or certain number of generations met)
Evaluate fitness
While (number of children=population size)
Select parents
Apply evolutionary operators to create children
End while
Step 5. End While
Step 6. Return Best solution
Step 7. End

VI. EVALUATION AND RESULTS

We have designed a Multiclass Classifier to demonstrate our results. We have used Java 6.0 as a front end tool and Sql as a back end tool to develop our project. We have used 3 real data sets for training and validating our methodology. These are Car Evaluation, Breast Cancer, and University. Table I gives a brief description about all the data sets used.

Table 1. Datasets

No. of Datasets	Number of Instances:	Number of Attributes:
Car Evaluation	1728	6
Breast Cancer	286	9
University	285	17

A. Parameters:

These are some parameter for different operations.

Table 2. Describes the common parameters used for all the data sets

Parameters	Values
Probability of Crossover Operation	80-90%
Probability of Reproduction Operation	15%
Probability of Mutation Operation	20-25%
Population Size	150-500
Minimum Tree Depth	3
Maximum Tree Depth	5
Number of Generations	25-60

Table 3. Experimentation Performed on Different Dataset by varying the number of training samples

Sr.No.	Training Error	Testing error	Generalization Error
1	3.7896	3.564	3.569
2	4.2345	3.456	2.265
3	4.5689	3.325	2.554
4	5.1234	3.112	2.545
5	5.2685	3.110	1.987

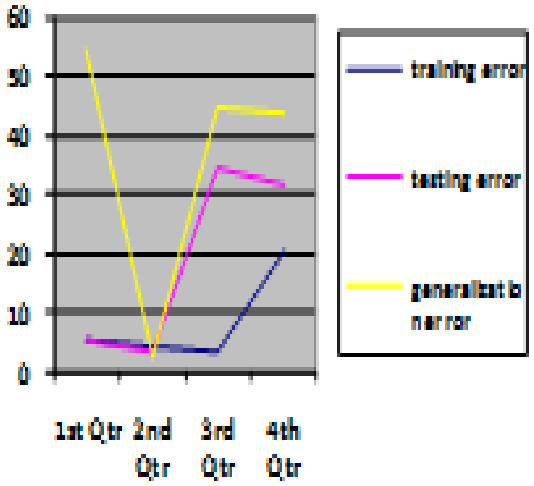

Figure.4. Numbers of training, Testing, Generalization error

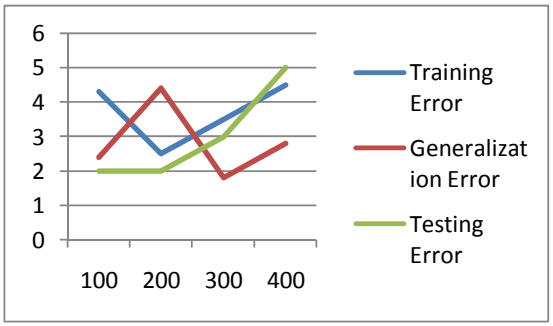

Figure.5. Numbers of training samples versus percentage error

VII. CONCLUSIONS AND FUTURE WORKS

In this paper we have presented a GP Design for explicit rule extraction. The system has been tested on publicly available databases. We have compared our system with neural network-based approaches and with other GP-based techniques. Experimental results have demonstrated the effectiveness of the approach proposed in providing the user with compact and comprehensible classification rules, and its robustness in terms of low standard deviation. Future work will include the application of the system proposed to other real-world datasets in order to further validate the promising results reported in the present paper, as for example in computational archaeology. Moreover, another interesting task to face will be the unsupervised data mining in which the goal is to discover rules that predict a value of a goal attribute which, unlike classification, is not chosen a priori.

VIII. REFERENCE

[1] Peter A. Whig ham, Member IEEE, and Grant Dick, Member IEEE, "Implicitly Controlling Bloat in Genetic Programming," IEEE Transactions on Evolutionary Computation, Vol. 14, No. 2, APRIL 2010.

[2] W. B. Langdon and R. Poli, "Fitness causes bloat: Mutation, "in Proc.Theory Application Evolutionary Comput. (ET'97), London, U.K.: University College London, 1997, pp. 59–77.

[3] W. Banzhaf and W. Langdon, "Some considerations on the reason for bloat, "Genetic, Programming Evolvable Mach., vol. 3, no. 1, pp. 81–91, 2002.

[4] R. Poli, "A simple but theoretically- motivated method to control bloat in genetic programming," in Proc. Genetic Programming (EuroGP '03) vol. 2610. Essex: Springer-Verlag, Apr. 14–16, 2003, pp. 204– 217.

[5] H. Stringer and A. Wu, "Bloat is unnatural: An analysis of changes invariable chromosome length absent selection pressure,"

[6] University of Central Florida, Tech. Rep. CS-TR-04-01, 2004.

[7] H. Stringer and A. Wu, "Winnowing wheat from chaff: The chunking GA," in Proc. Genet. Evol. Comput. (GECCO' 04) Part II, vol.

[8] 3103. Seattle, WA: Springer-Verlag, Jun. 26–30, 2004, pp. 198-209.

[9] C. Skinner, P. J. Riddle, and C. Triggs,"Mathematics prevents bloat," in Proc. 2005 IEEE Congr. Evol. Comput. vol. 1. Edinburgh, U.K.: IEEE Press, Sep.2–5, 2005, pp. 390–395.

[10] S.Bleuler,M.Brack,L.Thiele,andE.Zitzler, "Multiobjective Genetic Programming: Reducing bloat using SPEA2," in Proc. 2001 Congr .Evol. Comput. (CEC '01), Piscataway, NJ: IEEE Press, 2001, pp.536–543.

[11] Chaudhari, N.S., Purohit, A., Tiwari, A, "A multiclass classifier using Genetic Programming" Control, Automation, Robotics and Vision, ICARCV 2008.10th International Conference [1] Peter A. Whig ham, Member IEEE, and Grant Dick, Member IEEE, "Implicitly Controlling Bloat in Genetic Programming,"IEEE Transactions on Evolutionary Computation, Vol. 14, No. 2, APRIL 2010.

[12] D J Nagendra Kumar, Suresh Chandra Satapathy, J V R Murthy "a scalable genetic programming multi-class ensemble classifier"

[13] World Congress on Nature Biologically Inspired Computing NaBIC 2009 (2009).

[14] D.Agnelli, A. Bollini, and L. Lombardi, "Image classification: an evolutionary approach," Pattern Recognit. Lett., vol.23,pp. 303–309, 2002.

[15] S. Hettich and S. D. Bay. The UCI KDD Archive [http://kdd/ics/uci/edu]. Irvine, CA: University of California, Dept. of Information and Comp. Science, 1999.

[16] E. Bernado-Mansilla and J. M. Garrell-Guiu. Accuracy-based learning classifier systems: Models, analysis and applications to classification tasks. Evolutionary Computation, 11(3):209– 238, 2003.

[17] M. Brameier and W. Banzhaf. Evolving teams of predictors with linear genetic programming. Genetic Programming and Evolvable Machines, 2(4):381–407, 2001.

[18] M. Brameier and W. Banzhaf. Linear Genetic Programming. Springer, Genetic and Evolutionary Computation Series, 2007.

[19] E. D. De Jong and J. B. Pollack. Ideal evaluation from coevolution. Evolutionary Computation, 12:159–192, 2004.

[20] C. Elkan. Results of the KDD'99 classifier learning. SIGKDD Explorations, 1(2):63–64, 2000.

[21] S. G. Ficici and J. B. Pollack. Pareto optimality in coevolutionary learning. In Proceedings of the 6th European Conference on Advances in Artificial Life, pages 316–325, 2001.