RESEARCH ARTICLE OPEN ACCESS

Hybrid Genetic Algorithm for Extracting Rules Considering the Dataset Complexity

John Tsiligaridis
Department of Math and Computer Science
Heritage University, Toppenish
WA, USA

ABSTRACT
This work provides a method for rule extraction using hybrid GA, the GANN, which is a combination of Genetic Algorithm (GA) and a Neural Network (NN). In addition, a rule extraction algorithm based on weight (REAW), and a probabilistic Decision Tree (DT) algorithm focusing on large frequency classes (DTPL) are also developed. A NN is composed of many neurons that are linked together according to a specific network topology. A GA, which is suitable for non-linear problems, is employed to define the network topology which leads to accurate extracted rules. The fitness function is based on the predictive accuracy of rules to be extracted from the network topology. After the end of GANN process, which provides the NN parameters an algorithm (REAW) ,based on weights, is developed for extraction rules. REAW is running after the NN is trained with GA. The rules are extracted from the population with the fittest chromosomes. The complexity of a dataset depends on many parameters. For sets with low complexity, the DTPL can outperform the proposed method, in terms of accuracy, based on the cooperation of NN with GA. Simulation results with different complexity data sets are provided.
Keywords :— Genetic Algorithm, Decision Tree, Neural Network, Data Mining.

I. INTRODUCTION

DTs are one of the most popular techniques of Data Mining. These are used for the tree based classification [1],[2],[12]. NNs are widely used for classification problems. As the network size increases the algorithms associated with the production of rule sets tend to have higher complexity. The use of GAs will optimize the network topology leading to shorten the search space. A GA [8],[11] is a class of adaptive stochastic optimization algorithms involving search and optimization. The GA uses the chromosomes and map them directly onto intelligible rules (phenotype)[3]. The chromosome is easily converted into "if…then" rules including the attached weighted. The purpose of this work is to combine the idea of using a GA to evolve a network topology with the idea of extracting rules from a NN. For the discovery and extraction of the rules methods based on the activation values have been developed [2],[7],[10],[13]. A set of discrete activation values can define the input values and the hidden nodes activation values first, and the output values and the hidden nodes activation values afterwards. The rules are generated considering the output values after the enumeration of the discretized hidden node activation values. Finally, the step of the enumeration of the corresponding input values follows. More details in [5]. An algorithm for extraction of rules based on weights (REAW) is developed. Single and multiple condition rules are extracted considering the two types of weights; the input weights (between input and the hidden units) and the output weights (between hidden and the output units). REAW can also be used for discovering strong rules (larger support and confidence). Two types of datasets, the low and high complexity, based on parameters' number (size of instances etc), are considered. Experiments show the different abilities of the proposed algorithms on these two types.

II. DTPL

The DTPL can be created in the following phases:
Phase 1: Discover the root (from all the attributes)

$$P(EA) = \sum_{C} \sum_{A} p(A) * p\left(\frac{C}{A}\right)$$

where A: the attributes of the tuples and C
the classes (attribute test).
MP = max (P(EA)) //max attribute test criterion
Phase 2: Split the data into smaller subsets, so that the partition to be as pure as possible using the same formula. The measure of nodes impurity is the MP. Continue until the end of the attributes.
The large frequency classes (DTPL) are also extracted. The CEB criterion eliminate redundant branches. Most of the decision trees inducers require rebuilding the tree from scratch for reflecting new data that has become available.
For an attribute (attr1) with value v1, if there are tuples from attr2 that have all the values in relation with v1 (of attr1) then the attr2 is named as: *do n't care* attribute. The criterion of elimination of Branch (CEB) is used to avoid repetition and replication and it is given by:

$$P_{CEB} = p(A_1 = a_1, \dots, A_{|A|} = a_{|A|} | C = c_i) = \\ = \prod_{i=1}^{|A|} p(A_i = a_i | C = c_i)$$

If $P_{CEB} = 0$, between two attributes (A1, A2) then A2 is don't care attribute. The CEB criterion is valid when $P_{CEB} \neq 0$

Theorem: the CEB criterion can determine the existence of a small DT with the best accuracy (100%, or complete) avoiding repetitions and replications. *Proof:* When the CEB criterion is valid, it discourages the repetition.

III. NN

A NN [2],[7],[10], is a collection of units that are connected in some pattern to allow communication between the units. The test data set and the training data set should be disjoint (so that test data are not used during training). For the NN in the input layer the neurons correspond to prediction attribute values of the data set, and the output layer represents the predicted classes.

The specification of a typical neural network model requires the choice of the type of inputs, the number of hidden units, the number of hidden layers and the connection structure between the inputs and the output layers

IV. GA

If there are n input links with binary values, a 2^n different patterns can be produced. A Genetic Algorithm (GA) [3],[4],[8],[9] is used as a search algorithm for the rule selection process and refines the rules in order to provide better accuracy and coverage. For this purpose, a scoring function, fitness, is created so that GA can provide rules with higher classification accuracy and coverage. Two are the components of a GA: the genetic representation (encoding) and the fitness function. The operations: selection, crossover, mutation are used in order to converge the GA to a solution.

The fitness function has high capability to remove the noise rules. The GA is a classifier, that searches for the best chromosome (with the highest score value) using the fitness function. A pseudo-code for this algorithm is:

1. creation of the initial population.
2. while (! solution)

(a) Evaluate the fitness of all the chromosomes of the population.

(b) The best chromosomes will be selected to reproduce, using mutation and crossover.

(c) Substitute the worsts chromosomes of the previous generation by the newly produced chromosomes [3].

V. GANN

For the network topology a set of parameters (# of hidden layers, # of neurons per layer) needs to be tuned, instead of brute-force searching all the combinations. To this end, GA can help to "jump" from one combination to another. In this way, the search space can be "explored" for potential candidates. The application of GA to NN makes a hybrid GA (GANN) where the weights of the NN are calculated using GA approach. From all the search spaces of all the possible weights, the genetic algorithm will generate new points of the possible solution [2]. The GANN works with different datasets. For each dataset different weights and hidden nodes are discovered. The algorithm works with different number of iterations or according to a predefined stop condition (accuracy threshold) and defines the NN topology. After that the NN is able to discover classification rules. The hyperbolic tangent function is used as activation function between the input node and the hidden node and the sigmoid activation function between the hidden and the output function.

The pseudocode of GANN is as follows:

1.Initial conditions for NN topology
(number of inputs, numbers of hidden nodes, number of outputs)
2.Create initial population for GA
3.Compute weights for each chromosome
4.Computer fitness for each chromosome (MSE)
5. Test for exit (criterion)
if (not) GA creates a new population
used of: selection, crossover, mutation operators
go to step 2 (for the creation of the next population)
else:
the population with the best fitness chromosomes have been selected
the best weights will be used for the classification purpose

The exit criterion can be the number of iterations (generations) or to continue running until the mean square error (MSE) ≤ 0.001. The fitness is computed for all the chromosomes of the population and the best fit value chromosomes replace the worst fit ones. The process of selection, crossover and mutation follows and generates the next population. This process continues until many of the chromosomes converge to the same fitness value (convergence criterion). The final converged population with the best fit chromosomes has the optimized connection weights for the NN. Fitness is given by (Ci) = 1/MSE for each chromosome of the population, where MSE is computed at the output layer using the sigmoid activation function.

VI. REAW

Considering the Data Mining approach, the knowledge held by the weights in the NNs interconnections is their strength. The extracted classification rules have the antecedent part ("if") containing the input values with the conjunction condition of the attribute values and the ("then") part consequent with the class value. After the GANN, the topology with the weights of the network is saved and is used for he rule extraction process. Due to the continuous activation values of the hidden unit, the rules are not extracted directly from the NN. The discretization of these values can be achieved by using a clustering algorithm (or binning method, [1]) and in this way the activation values are discretized into a number of discrete values. In addition the discrete values are discovered for each hidden unit in respect to keeping the accuracy of the training set. The number of the output units' outcomes is dictated by the number of activation values of each hidden unit. The rules are generated by the input and output weights. REAW searches for the weights that can maintain the accuracy of the training set.

For the accuracy the confusion matrix [13] referred to two-class problem is considered. For encoding the Michigan Approach [14] is used.

A conjunction of at most (n-1) attributes consist the antecedent of a rule. The consequent contains the decision as a single term having the goal attribute.

Single and multiple condition rules can be extracted with REAW. For the single condition rule extraction, a rule template method can be created ("if gene1 (weight between input unit and hidden unit) then class"). Two genes can be considered. The one gene stands for a node in the input layer and the other gene stands for a node of the hidden layer. The consequence of the rule comes from the activation on the particular input node. The fitness can be computed as the absolute summation of the weights from input to hidden unit and from hidden unit to output considering also the accuracy. For example, for a dataset, the list [4,2] represents the input layer is the 4th unit and the 2nd unit in the hidden layer. The fitness of the chromosome will be : 4.3230 (-2.0567 from the input unit to hidden unit and -2.2663 from hidden unit to output). The multiple condition rule extraction is an extension of the single condition extraction tuples. The process starts first from the part related to the hidden layer nodes and the output (part1) and continue with the input and the corresponding hidden layer nodes (part2).

The REAW has the following phases.

input: the input weights (w_{ij}, between i inputs and j hidden units), the output weights (w'_{jk} ,between j hidden units and k output units), thr:
is the minimum acceptable value of accuracy
output: rules
variables: n input units and m hidden units, r_m : rules $r_1, r_2..r_t$
//compute the weights (w, w')
//A. for a single condition rule
for each hidden unit
  for each discretized activation value
    find the weights (w'_{jk}) for a class output (part1)
  for each weight (w'_{jk})
    find the weight (w_{ij}) (part2)
    create the rule r_m with fitness
      sum ($w_{ij} + w'_{jk}$), m=1..t (# rules)
    extract the rule with max (sum) and
      accuracy < thr

//B. for multiple condition rules
define the weights (w'_{jk}) for a class output (part1)
for each weight (w'_{jk})
  find the weights (w_{rj}), where r=2..g
    (r= # selected of attribute,
    g=total # of attributes) (part2)
  create the rule r_m with fitness:
    sum ($w_{rj} + w'_{jk}$),
  extract the rule with sum ≤ max (sum)
    and accuracy ≥ thr

The rules can be created by taking the maximum fitness values, considering also to maintain the accuracy for the training set. A threshold value is used in order to have the desired accuracy. If the accuracy falls under the desired value with the maximum summation (ideal condition) of the weights, then there are two cases. First, finding weights with lower than the maximum summation providing also the desired accuracy. A sorting algorithm for the weights is used for that purpose. Second, changing the threshold to lower acceptable values. The number of rules that are extracted from each population depends on the complexity of data and the network structure. Most important rules can come from most fit chromosomes.

The REAW for the IRIS dataset has given the rules (single and multiple condition): R1: if petal length ≤ 1.9 then setosa
R2: if petal width > 1.6 then virginica
R3: if petal length > 1.9 and petal width ≤ 1.6 then versicolor

For the ZOO dataset the multiple condition rules:
R1: if domestic =0 and aquatic =1 then class=3
R2: if breathes=1 and domestic=1 then class=6
R3: if fins=1 and cat-size=1 then class=1

VII. COMPLEXITY

The complexity of rule discovery arises from the fact that the number of possible rules in a given dataset all of which must be examined in exhaustive rule discovery, may be very large.

It can be seen [6] that the number of possible rules in a given dataset is approximately as shown below:

$$r\left(\prod_{i=1}^{n}(k_i+1)-1\right)$$

r is the # of outcomes , n is # of attributes classes and k_i is the number of values of each i attribute.

The complexity of a ruleset depends on the size of a dataset, the number of the attributes, and the number of the cases. Experiments for the different behavior of the dataset are presented.

VIII. SIMULATION

Simulation is based on various experiments.

1. DTPL vs GA_NN : DTPL extract rules for iris dataset (low complexity) faster than GANN (evolution for 500 iterations). It takes more time for GANN even for low complexity data.

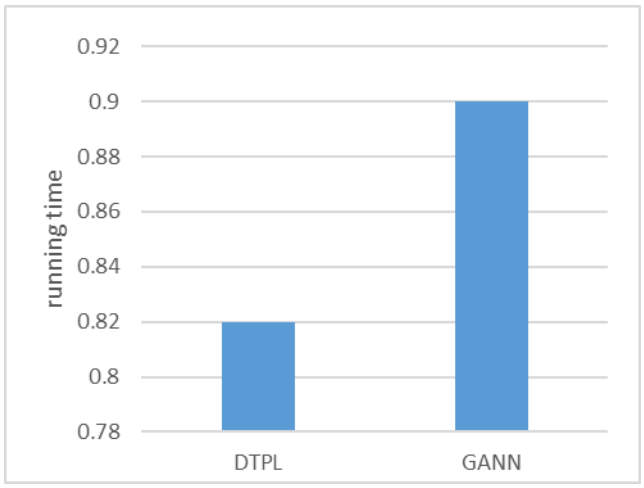

Fig. 1 DTPL vs GANN

2.The inverse error: The inverse error is examined while the GANN works and define the NN parameters. The inverse error of the population (population average error -1/MSE-) increases as the MSE becomes smaller until the GANN terminates. The termination condition is the number of the population generations from GANN. At that time GANN is able to define the NN parameters which will be used for classification of the datasets. In Fig.1 there is the evolution of the population inverse error.

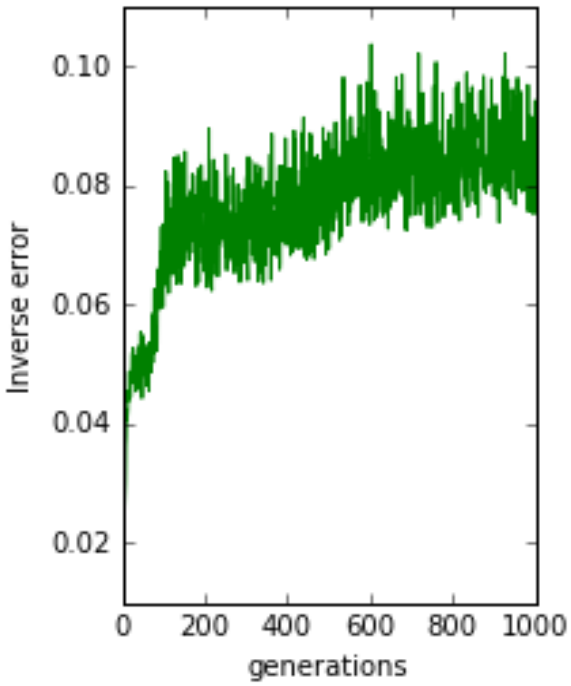

Fig.2 Evolution of population inverse error

3. *Accuracy:* Fig.3 shows the training time error for breast cancer dataset. After the preparation for the NN parameters It was observed that the training set gives good values for the accuracy.

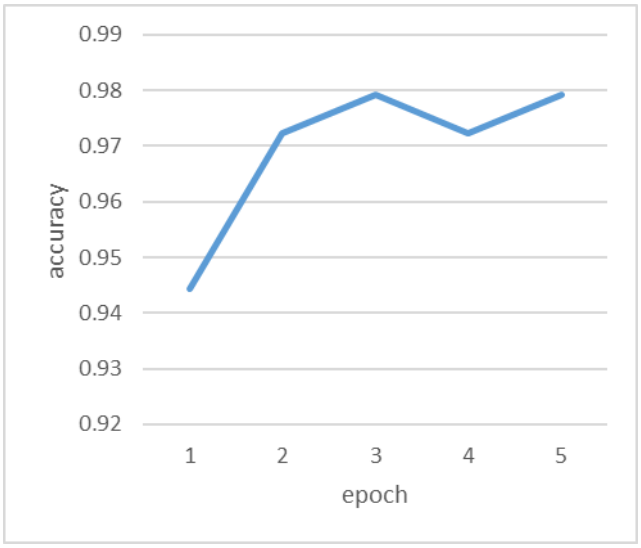

Fig. 3 Accuracy for the breast cancer data

4. *The complexity:* The complexity of a dataset depends on the number of instances, the number of attributes, the number of values for each attribute and the number of the class labels. It is shown that the GANN can extract rules with high predictive accuracy. The accuracy rate of the discovered rules was 97.4% and 98% for Car Evaluation and Breast Cancer respectively. GANN works fine for the rule discovering no matter of the how many attributes with their number of values

and classes' number is included in the datasets. For example, Breast Cancer, 9 input attributes with a number of values and the Car Evaluation with 6 input variables and 4 output cases.

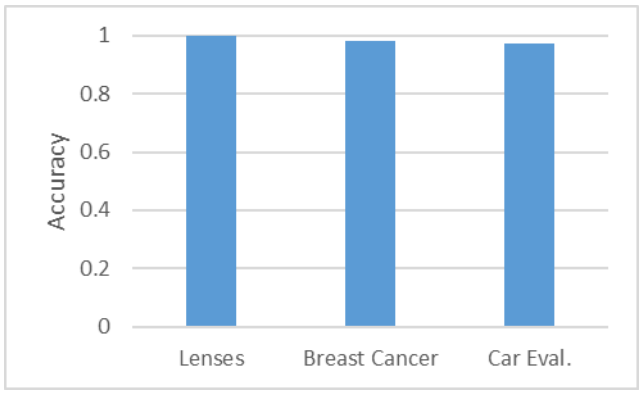

Fig. 4 Accuracy of rules with GANN after 10 generations

IX. CONCLUSIONS

The discovery of the NN parameters play significant role for the extraction rules from dataset. The hybrid GA finds out the suitable parameters for NN design. DTPL has superiority over the GANN due to the low complexity data.

REAW can extract rules using the two types of weights. REAW can work also for discovering many rules (significant or not) considering the different types of weight's combination for different complexity datasets.

For higher complexity data, GANN can provide good accuracy since it can define the correct parameters of NN that minimize the MSE. Future work will be the use of NNs for classification purposes.

REFERENCES

[1] J. Han, M. Kamber, J. Pei, "Data Mining Concepts and Techniques", Morgan Kaufman 3ed ,2012

[2] M. Karntardzic, "Data Mining: Concepts, Models, Methods, and Algorithms", IEEE Press, 2003

[3] D. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Pearson, 2002.

[4] S. Perez, "Apply genetic algorithm to the learning phase of a neural network ", [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc / /download?doi=10.1.1.187.5220&rep=rep1&type=pdf

[5] H. Lin, S. Tan, (1995),"X2R: A Fast Generator", Int. Conference on Systems, Man and Cybernetics", Vancouver, Canada,1995

[6] D. McSherry, "Attribute-value distribution as a strategy for increasing the efficiency of data mining", IEEE Colloquium Knowledge Discovery and Data Mining, (Digest No. 1998/310), June 1998.

[7] S. Haykin, "Neural networks: A Comprehensive Foundation", Prentice Hall, 2e

[8] A. Freitas, "Data Mining and knowledge Discovery with Evolutionary Algorithms", Springer-Verlag, Berlin Heidelberg,2002.

[9] R. Carvalho, A. Freitas, "A gentic-algorithm for discovering small-disjunct rules in data mining", Applied Soft Computing, vol.2, 2002, pp. 75-88.

[10] M. Smith, "Neural Networks for Statistical Modeling, Nan Nostrand Reinhold, 1993.

[11] H. Bhasim, S. Bhatia, "Application of Genetic Algorithms in Machine Learning", IJCSIT International Journal of Computer Science and Information Technologies, Vol.2 (5), 2011, pp. 2412-2415.

[12] M. Bramer," Principles of Data Mining", Springer-Verlag, London Limited, 2007. [13] H. I. Witten, E. Frank, "Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations", 2nd edition. Morgan Kaufmann,2005

[13] A. Freitas," Data Mining and Knowledge Discovery with Evolutionary Algorithms", Springer, 2002.