A NEW INCREMENTAL GENETIC ALGORITHM BASED CLASSIFICATION MODEL TO MINE DATA WITH CONCEPT DRIFT

[1]P.VIVEKANANDAN , [2] DR. R. NEDUNCHEZHIAN,

[1]Department of Computer Science and Engineering
Park College of Engineering and Technology
Coimbatore, India
[2]Department of Computer Science and Engineering
KalaignarKarunanidhi Institute of Technology
Coimbatore, India

ABSTRACT

In a database the data concepts changes over time and this is called concept drift. Genetic algorithm is widely used for mining classification rules. If the data set is of three or four years old, the mined rules may not reflect the current concept due to concept drift. Applying the incremental genetic (IGA) algorithm in a batch mode can mine accurate rules reflecting the current concept. But applying genetic algorithm without monitoring for a change in an incremental manner repeatedly on arriving data will result in an unnecessary increase in the learning cost. There is also another problem, Due to change in the data distribution some of the rules which are generated may be lost when we apply genetic algorithm in an incremental fashion. In this paper a new incremental genetic algorithm is proposed to rectify the above problems. The New IGA applies the Genetic algorithm iteration step only when required, so that learning cost may be reduced. The new method also keeps track of the rules which are generated earlier and which would have been lost due to change in data distribution. In the proposed method each record of the incoming dataset is monitored. If they are correctly classified they are dropped and misclassified records are added to a window. When the window is full, the genetic algorithm is applied to the records in the window and new rules are generated based only on the misclassified examples and on the examples of new classes. The invalid rules are replaced with the newly generated valid rules. The new method ensures that the next iteration of genetic algorithm is called only when there is a concept drift or when there is a change in the data distribution and sufficient number of records is available. This will reduce the learning cost particularly when there is no concept drift or when there is a slow drift and also ensures that no rule is lost due to change in data distribution.

Key words: *Classification, Incremental Genetic Algorithm, Concept Drif*

1. INTRODUCTION:

Genetic Algorithm (GA) is a stochastic search method which has been inspired by the data mining community. It is widely used in discovering classification rules [2, 3, 4, 5, and 6]. The rules that GA finds are more general because of its global nature. GA shows great promise in complex domains because it operates in an iterative improvement fashion where search is probabilistically concentrated towards regions of the model representation space that have been found to produce a good classification behavior.

Today's data like stock market, customer buying are liable to concept drift. The target concept changes due to changes in the under lying context [7]. This phenomenon is called concept drift. Due to this the classification models built using genetic algorithm on three to four years data will not reflect the current concept. Incremental genetic algorithm (IGA) is proposed by Huai li et al [1] to rectify this problem. Incremental genetic algorithm will produce accurate rules reflecting the latest concept but with a considerable learning cost overhead because it applies IGA in a repeated manner without analyzing the nature of the concept drift. When we look into the real world data, all the rules will not change from valid to invalid due to the concept drift and also even there may not be any concept drift at all. The concept drift can also occur at any time and it may be a slow or a fast drift. So Executing genetic algorithm [1] for each batch in an incremental fashion without analyzing the data for concept drift is an unnecessary act particularly when there is no concept drift. The new method proposed in this paper ensures that incremental genetic algorithm is applied only when there is a need. This will prevent unnecessary call to genetic algorithm particularly when there is no concept drift.

The paper is organized as follows, section 1 describes about previous work, section 2 describes the proposed method, Section 3 describes the experimental results and section 4 concludes the paper.

1. Previous work

Traditional classification methods like C4.5 are designed to deal with static data [14]. Subsequently many new non GA based methods have been proposed to deal with concept drifts [9, 10, 11, 12, and 13]. An incremental GA was proposed by Gaun et al [8] which updates the rules based on the new data. Due to the arrival of new data or new attribute or class, the classification model may change. So to deal with this the author proposes an incremental based GA. As discussed earlier Huai li et al [1] proposes a memory based incremental genetic algorithm to deal with the concept drift. They make an assumption that the new training data pass through a fixed-size window at a steady rate. When the window is full, genetic algorithm is applied to determine the set of classification rules. Old instance of the window are replaced with the arriving new training samples. Once all the original samples have been replaced by new samples, the GA is called to determine the new set of best classification rules. This procedure is repeated sequentially as long as learning is required. Their GA utilizes a memory-based random immigrant module, in which the initial population pool of the GA applied at each stage of the incremental learning process comprises a mix of best solutions obtained in the previous stage and an appropriate number of random immigrants. Their results demonstrate that IGA achieves a comparable classification performance to that obtained using existing incremental and non-incremental methods.

In practical data sets, concept drift may or may not be present. If present, it may occur at any time (i.e.) it may appear in the second window or at the last window. All the records will not be changed due to concept drift and so all the rules will not go invalid. The concept drift may be very slow and requires certain time to become effective. So repeatedly applying IGA without monitoring the nature of concept drift will increase the learning cost significantly. There is also another important problem of lose of some of the old rules in IGA which affects the accuracy of the model. If the underlying data distribution changes and if some of the data does not reappear, we may also lose some of the rules due to the incremental nature of genetic algorithm.

For example, consider the data in the given three consecutive blocks described in Table 1, Table2 and Table 3 respectively.

Table 1 Example Block1 data

RID	Record	Class
1	$a_1b_1\,c_1$	c_1
2	$a_1b_1c_2$	c_1
3	$a_1b_3c_3$	c_2
4	$a_3b_3c_3$	c_2
5	$a_3b_1c_1$	c_3
6	$a_3b_2c_1$	c_3
7	$a_3b_2c_1$	c_3
8	$a_1b_1c_1$	c_1

Table 2 Example Block2 data

RID	Record	Class
1	$a_1\,b_3c_3$	c_2
2	$a_1\,b_1c_1$	c_1
3	a_1b1c3	c1
4	$a_3\,b_3c_3$	c_2
5	$a_1\,b_3c_3$	c_2
6	$a_1\,b_1c_1$	c_3
7	$a_3\,b_1c_1$	c_3
8	$a_1\,b_3c_3$	c_2

Table 3 Example Block 3 data

RID	Record	Class
1	$a_1 b_3 c_3$	c_2
2	$a_2 b_3 c_3$	c_2
3	$a_3 b_1 c_1$	c_3
4	$a_2 b_1 c_1$	c_3
5	$a_3 b_2 c_1$	c_3
6	$a_1 b_2 c_3$	c_4
7	$a_2 b_2 c_3$	c_4
8	$a_1 b_3 c_3$	c_2

IGA method is used to mine the classification rules for the given three blocks of data. Let the window size be ten so that each block will fit to a window. According to IGA method, a fixed sized window is made to slide over the examples and the genetic algorithm is called whenever the window is full. The rules generated in one iteration forms the initial population for the next iteration. So when the IGA is applied for the above example there are three iterations of genetic algorithm for the given three blocks of data.

The rules mined during first iteration by using first block are

$a_1b_1 \rightarrow c_1,\ b_3c_3 \rightarrow c_2,\ b_1c_1 \rightarrow c_3$ (1)

Considering the next block of data and the rules mined are

$a_1b_1 \rightarrow c_1,\ b_3c_3 \rightarrow c_2,\ b_1c_1 \rightarrow c_3$ (2)

In the third iteration the rules mined are

$b_2c_3 \rightarrow c_4,\ b_3c_3 \rightarrow c_2,\ b_1c_1 \rightarrow c_3$ (3)

When examining the above three rule sets (Equations 1, 2 and 3), three important characteristics of the rules generated can be identified. First, there is no difference between the rules in the rule set one and rule set two. This similarity is due to non existence of concept drift in the second block. So applying genetic algorithm during second iteration is an unnecessary act because it has not changed the rule set. Second, When Applying IGA to the third block, only one rule has changed and there is no change in the remaining rules when compared to the rules in rule set two. There are only two misclassified records in the third block and remaining records are classified correctly by the previous rule set. So we can notice that the correctly classified examples does not change the rule set significantly. Third, In the third iteration rule set, rules of class c1 does not appear because there is a change in data distribution when we compare data in the second and the third block . This proves that we may also lose some of the rules due to change in the data distribution even though there is no concept drift.

From the above discussion following conclusions can be made

1. If there is no concept drift IGA need not be applied.
2. Correctly classified records can be dropped since they do not change the rule set significantly,
3. Remembering all the rules from the first iteration of IGA and replacing only the invalid rules in the subsequent iterations will prevent lose of rules due to change in the data distribution .

The method proposed in this paper is based on the above three points. It is not necessary to apply the IGA repeatedly batch after batch of samples. Instead it can be applied only after detecting concept drift and also only to the misclassified records. This will greatly reduce the computational overhead particularly when there is no concept drift or when there is a slow drift.

The core genetic algorithm of the method proposed in this paper is similar to the method proposed by Huai li et al[1]. In Genetic Algorithm the potential solutions are encoded as chromosomes. This is called population. The population is processed using heuristic search, selection, crossover, mutation and evaluation operation such that the best solution gradually emerges. Section 1.1 describes the encoding process and Section 1.7 describes the basic architecture of the Genetic algorithm. Rest of the sub sections describes each of the other processes involved.

1.1Encoding

Let there be n input attributes and one target attribute. The chromosome contains n+ 1 gene. Each gene in the chromosome represents an input attribute and the last gene represents the target attribute. Suppose if an attribute can take m different values it is encoded by m+1 binary bits ,m bits for each value of the attribute .and one bit to indicate whether or not the attribute forms part of classification rule or not. Each chromosome represents a rule of classification. A population is a set of chromosomes representing the best solution at that instant. Initial population is created randomly.

Consider a set of a training sample with three attributes A1, A2, A3 and a target attribute A4. They take 3,4,2,3 values respectively. Table 4 represents the chromosome coding and its meaning. Since Flag bit of A2 is Zero, A2 does not take part in the rule.

Table 4 Chromosome Coding

Attribute	A1	A2	A3	A4
Code	1001	00001	110	100
Rule	IF A1= Value3 and A3= Value1 Then A4= Value1			

1.2 Heuristic search module

Two different heuristic search strategies are applied after crossover. In the first strategy, the value which appears most frequently for each attribute is identified and replaced within each chromosome by a randomly selected value. If the value of the fitness function is improved, the randomly selected value is accepted as the new attribute value; else the original value is restored. In the second search strategy, one attribute in the chromosome string is chosen at random and its first bit is inspected if it is “1” it is changed to “0” so that it is discarded from the rule. The fitness of the resulting chromosome is then re-evaluated. If the fitness value is found to improve, then the attribute is discarded from the classification rule, else it is retained.

1.3 Crossover

Two point crossover method is employed. Two chromosomes are selected randomly according to the crossover probability for reproduction using roulette wheel selection method [1]. Two cutting points are randomly selected between the pair of chromosomes. The bit string between cutting points are simply exchanged between the two chromosomes.

1.4 Mutation

A single chromosome within the population is chosen in random and a bit within the population is also chosen at random and its value is flipped. The mutation probability is dynamic and depends upon the average fitness value of the chromosomes according to the equation 4. It is initialized to 0.5 before the first generation begins.

Mut-rate=1-avgfit/2 (4)

1.5 Selection

Best n elite chromosomes are chosen from both the parent and child for the next generation population pool. Remaining chromosomes are chosen from the child population to form the initial population pool for the next generation. The GA process is continued and stopped such that there is no significant increase in the fitness function for say some m generations. It can also be stopped after specific number of generation has been elapsed.

1.6. Fitness Function calculation

The quality of each of the chromosomes within the population is evaluated using the fitness function represented by Equation (7); this fitness function comprises two components, namely a sensitivity term (see Equation(5)) and a specificity term (see Equation (6)).

Sensitivity=tp/tp+fn (5)

Specificity=tn/tn+fp (6)

Fintness=tp/(tp+A.fn) * tn/(tn+B.fp) (7)

$0.2<A<2$ $1<B<20$

In the equation *tp* denotes true positive, *fp* denotes false positive, *tn* denotes true negative, and *fn* denotes false negative . A and B are user-defined parameters which determine the rate of convergence of the solution procedure and are determined experimentally in accordance with the requirements

If the rules are of form X → Y Then

1. True positive (*tp*): the actual class is Y and the predicted class is also Y.
2. False positive (*fp*): the actual class is Y, but the predicted class is not Y.
3. True negative (*tn*): the actual class is not Y and the predicted class is also not Y.
4. False negative (*fn*): the actual class is not Y, but
The predicted class is Y.

1.7 Genetic Algorithm

1. [Start] Generate random population of n chromosomes (Section 1.1)

2. [Fitness] Evaluate the fitness of each chromosome in the population (section 1.6).
3. [New population] Create a new population by repeating following steps until the new population is complete
    1. [Selection] Select two parent chromosomes from a population according to their fitness (section1.3).
    2. [Crossover] With a crossover probability cross over the parents to form new offspring (section 1.3).
    3. [Heuristic Function] Apply the heuristic Function to the new Chromosomes (Section 1.2)
    4. [Mutation] With a mutation probability mutate the new offspring (Section 1.4).
4. [Accepting] Select new population for a further run of the algorithm (Section 1.5).
5. [Test] If the end condition (section 1.5) is satisfied, stop, and return the current population as best solution, otherwise go to step 2.

2. PROPOSED METHOD

Two phases are there in the proposed method. First is the initialization phase A window Wi of size i is considered. The size of the window is fixed based on the number of samples required to mine accurate rules. The records from the incoming data set are copied one by one to the window. When the window is full the genetic algorithm is applied and the rules are mined.

Definition 1 (Misclassified Records). A Record is called as a misclassified record if 1) it is assigned a wrong class label by a rule or 2) it has no matching rule.

Second phase is the update phase. In this phase two counts called misclassified record count (MC_i) and correctly classified record count (CC_i) are maintained for each rule i. Both the counts are initialized to zero. Remaining Records are read one by one from the arriving data set and they are matched with the rules in the rule set. If a record is classified correctly by a rule in the rule set it is dropped and the corresponding rules CC is increased by one. If a record is misclassified (Definition 1) by the rule set, it is added to the window. Simultaneously The MC of the rule which misclassified it is increased by one. When the window becomes full next iteration of genetic algorithm is called again and new rules are generated. For each rule i in the old rule set, the percent of misclassified record count (PMC_i) is calculated using equation 8. The old rules whose PMC is greater than certain threshold are removed from the memory and replaced by the new rules. Next the CC and MC values of all the rules in the memory are again set to zero. The process is repeated until when there is no more examples.

$$PMCi = MCi / (CCi + MCi)*100 \qquad (8)$$

With the help of the above process, iteration step of the genetic algorithm is applied only when required. Testing of the accuracy of the generated rules is also performed simultaneously. This will ensures that only accurate rules will be generated and retained. Unnecessary application of the genetic algorithm step is also avoided. This will reduce computational overhead significantly.

2.1 The New IGA

1. [Initialization] Add the example records one by one to the Window of size i.

2. If window is full, apply the GA to the records in the window and generate new rules. Save the new rules in the memory.

3. Empty the window and create two counters MC and CC for each rule and initialize them to zero.

4. Read the next record in the incoming data set if it is classified correctly by some rule say j then increase the CC count of j by one and drop the record

5. Else if it is misclassified by some rule say m then increase the MC count of m by one and add the record to the window or else if the record belongs to a new class add it to the window

6. If the window is not full repeat the steps from 4 again

7. If the window is full Apply GA [Section 2.7] for the records in the window and generate new rules

8. Calculate the percentage of misclassified record count for the rules in the memory [Equation 4].

9. Remove the rules whose PMC is above the threshold from the memory and Add the new rules to the memory

10. Repeat again from step 2 if end condition is not reached.

3 . EXPERIMENTAL STUDY

The experiments were performed using the zoo data set downloaded from the University of California at Irvine-Machine Learning Repository (UCI) (15). The Zoo data set contains 101 training instances. There are 18 attributes and samples in the zoo data set belong to any one of the seven different classes. Out of the 18 attributes, one attribute indicates the name of the animal, two are numeric and the remaining are Boolean variables.

In the IGA simulations, the dataset was partitioned into four blocks of equal size. The population size was specified as 20 chromosomes and the fitness function parameters were specified as A=1 and B=1, respectively. Table 5 describes number of records of each class in all the four blocks. When IGA is applied the results are in par with results proposed by Huai li et al[1] and the results are tabulated in table 6. Next the proposed New IGA algorithm was applied and the results are tabulated in table 7. During the experiment study the size of the window is kept equivalent to the size of a block. So for the initialization phase the first block is used. The first block contains examples for classes 1, 2, 4, and 7 so the rules mined are for class 1, 2, 4 and 7 with an average 98% accuracy and requires 65 generations. Next in the update phase remaining records are read one by one, correctly classified record of classes 1, 2, 4 and 7 are dropped. Only misclassified records and records of new classes are added to the window. Even after reading all the examples the window is not full. Since there is no more records genetic algorithm is called again. New rules for the classes 3, 5 and 6 are generated in the second iteration. The average accuracy of all the rules after second iteration is 98%. Note here the genetic algorithm is applied only twice compared to four times in IGA. The average number of generations in the proposed method is also reduced considerably compared to IGA. Thus, this proves that learning cost is considerably less for the proposed method compared with the IGA.

Table 5 The Number of Training Examples of the given classes in each block

Class label	Block1	Block2	Block3	Block4	Total
1	11	12	12	5	40
2	5	4	5	6	20
3	0	0	1	4	5
4	5	2	3	3	13
5	0	2	1	1	4
6	0	5	1	2	8
7	4	1	2	4	11

Table 6 Results obtained when classifying Zoo using IGA

Simulation step	Evaluation generation	Accuracy
1	65	99%
2	54	92%
3	35	100%
4	25	100%

Table 7 Results obtained when classifying Zoo using proposed New IGA

Simulation step	Evaluation generation	Accuracy
1	65	99%
2	63	98%

4. CONCLUSION:

Incremental genetic algorithm is used to build accurate classification model when the example data is affected by concept drift. Two problems in the IGA methods are identified. First, incrementally applying genetic algorithm without monitoring concept drift will increase the learning cost. Next, if the data distribution changes, IGA may also forget some of the rules, if the data of that rules does not reappear. A new Incremental genetic algorithm is proposed to rectify the above two problems. With a sample data set, it has also been proved that the proposed method reduces the learning cost significantly when the data distribution is not uniform. In future studies, the effect of noise will be thoroughly investigated and the proposed algorithm will be modified accordingly to handle the effect of noise. The proposed method will also be rigorously tested with real data sets with concept drift to ascertain its efficiency.

REFERENCES:

[1]. I-hui li, I-en liao and Wei-zhi pang, "Mining classification rules in the presence of Concept drift with an incremental genetic Algorithm ", *journal of theoretical and applied information technology,* 2008.

[2]. K.A. De Jong., W.M. Spears and D.F. Gordon, "Using genetic algorithms for concept learning", *Machine Learning volume* 13, page(s) 161-188, 1993.

[3]. C.Z. Janikow, "A knowledge-intensive genetic algorithm for supervised learning", *Machine Learning volume* 13, page(s) 189-228, 1993.

[4]. D.P.Greene and S.F.Smith, "Competition-based induction of decision models from examples", *Machine Learning volume 13*, page(s) 229-257, 1993.

[5]. K.A. De Jong and W.M. Spears, "Using genetic algorithm for supervised concept learning", *Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence,* 1990.

[6]. E. Noda, A.A. Freitas and H.S. Lopes, "Discovering interesting prediction rule with a genetic algorithm ", Proceedings *of the 1999 Congress on Evolutionary Computation, Volume 2*, 1999

[7]. G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine Learning*, *volume 23*, no.1, page(s) 69–101, 1996.

[8]. S.U. Guan and F.ZhuCollard, "An incremental approach to genetic-algorithms based classification. Systems", *Man and Cybernetics, Part B, IEEE Transactions*, *volume 35*, no. 2, page(s) 227 – 239, 2005.

[9]. Jing Gao, Bolin Ding, Wei Fan, Jiawei Han,Philip S.Yu, "Classifying Data Streams with Skewed Class Distributions and Concept Drifts"**,** *IEEE Internet Computing, Special Issue on Data Stream Management*(IEEEIC),Nov/Dec. 2008, page(s)37-49, 2008.

[10]. A. Tsymbal, "The problem of concept drift: definitions and related work," *Department of Computer Science, Trinity College Dublin, Tech. Rep. TCD-CS-2004-15*, 2004.

[11]. H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page(s). 226–235, 2003.

[12]. G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams", *Proceedings of 7th ACM SIGKDD International Conference on Knowledge* and Data Mining, page(s) 97-106, 2001.

[13]. Mihai Lazarescu, Svetha Venkatesh and Hai Hung Bui (2004) "Using Multiple Windows to Track Concept Drift ", *Intelligent Data Analysis Journal, Volume 8 (1), 2004.*

[14]. Quinlan, J. R. "C4.5: program for machine learning", *Morgan Kaufmann.* 1992

[15]. http://www.ics.uci.edu/~mlearn / MLRepository.html