On Improving the Prediction Accuracy of a Decision Tree Using Genetic Algorithm

Md. Nasim Adnan[1](✉), Md. Zahidul Islam[1], and Md. Mostofa Akbar[2]

[1] School of Computing and Mathematics, Charles Sturt University, Bathurst, NSW 2795, Australia
{madnan,zislam}@csu.edu.au

[2] Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh
mostofa@cse.buet.ac.bd

Abstract. Decision trees are one of the most popular classifiers used in a wide range of real-world problems. Thus, it is very important to achieve higher prediction accuracy for decision trees. Most of the well-known decision tree induction algorithms used in practice are based on greedy approaches and hence do not consider conditional dependencies among the attributes. As a result, they may generate suboptimal solutions. In literature, often genetic programming-based (a complex variant of genetic algorithm) decision tree induction algorithms have been proposed to eliminate some of the problems of greedy approaches. However, none of the algorithms proposed so far can effectively address conditional dependencies among the attributes. In this paper, we propose a new, easy-to-implement genetic algorithm-based decision tree induction technique which is more likely to ascertain conditional dependencies among the attributes. An elaborate experimentation is conducted on thirty well known data sets from the UCI Machine Learning Repository in order to validate the effectiveness of the proposed technique.

Keywords: Decision tree · Genetic algorithm
Prediction accuracy · Knowledge discovery

1 Introduction

Data mining has entered into our day to day life; we now predict who would be the mayor of our town. This prediction is carried out by classifier(s) based on previously known information. In the same way, classifiers are used in business, science, education, medical, security and many other arena. As classifiers enter such influential and sensitive ambit, the importance of improving their prediction accuracy and knowledge discovery potential is paramount.

There are many different types classifiers in literature such as Artificial Neural Networks [40], Bayesian Classifiers [9], Nearest-Neighbor classifiers [19], Support Vector Machines [11] and Decision Trees [10,30,31]. Among them, the application domain of decision trees are considerably large as they can be readily

G. Gan et al. (Eds.): ADMA 2018, LNAI 11323, pp. 80–94, 2018.
https://doi.org/10.1007/978-3-030-05090-0_7

applied on data sets with categorical, numerical, high dimensional and redundant attributes [28]. More importantly, decision trees can express the patterns that exist in a data set into a set of logic rules (rules) that closely resembles human reasoning [27]. Also, decision trees require no domain knowledge for any parameter setting and therefore more appropriate for exploratory knowledge discovery [15]. Till date, C4.5 [30,31] remains to be one of the most accurate and popular decision tree induction algorithms [24]. In line with the above mentioned facts, we understand that any improvement beyond C4.5 can render significant influence over its large application domain.

Almost all popular decision tree induction algorithms such as C4.5 [30,31] and CART [10] follow the structure of Hunt's Concept Learning System (CLS) [17]. Hunt's CLS is in general a greedy (i.e. nonbacktracking) top-down partitioning strategy that attempts to secure "purer class distribution" in the succeeding partitions. Generally, a greedy strategy picks the locally best attribute that delivers the "purest class distribution" in succeeding partitions as the splitting attribute. However, this locally best attribute may not be the ultimate best attribute selected for that particular partition. The reason is: all impurity measures used for inducing decision trees assume that all non-class attributes are conditionally independent and hence ignore relationships that may have existed among some of the non-class attributes. As a result, greedy strategies may lead to the generation of suboptimal decision trees in applicable cases.

Building the optimal decision tree (in terms of prediction accuracy) by exhaustive search starts by placing each non-class attribute as the splitting attribute for each partition and generate a candidate decision tree from each combination of the splitting attributes. Then from all the generated decision trees, one candidate decision tree with the best prediction accuracy is recognized as the optimal decision tree. However, computing the optimal decision tree by exhaustive search is unrealisable with the increase of non-class attributes as the number of candidate decision trees grows exponentially.

In order to fix the problems of greedy approaches as well as avoid the computational burden of the exhaustive search, one-level look ahead strategies were developed. However, these strategies yielded larger and less accurate decision trees in many occasions [29]. As an obvious solution to these problems, Genetic Algorithm-based techniques can be applied that with high probability can generate optimal/near-optimal decision trees. Despite being computationally intensive, these algorithms are no longer exponential to the number of non-class attributes.

Genetic Algorithm (GA in this paper) is a class of computational framework inspired by evolution [38]. GA was first introduced by John H. Holland as an adaptation of natural evolution (***survival for the fittest***) in computing [16]. GA encodes a potential solution in a simple data structure called chromosome. Typically, the execution of a GA begins with a population of randomly defined chromosomes. Then GA iteratively moves forward by applying genetics-inspired operators/components such as crossover and mutation to create new population(s). Chromosomes of each population are evaluated and chromosomes representing better solution remain in the process to be given more chance to reproduce. The chromosome with the best solution so far is reported as the output of GA. Unlike greedy approaches that have a high chance of being stuck in a local optima for unidirectional search, GA performs a robust search in different directions of the solution space in order to find the optimal/near optimal solution [4].

The size of the solution space can be comparable to the size of an ensemble of decision trees. An ensemble of decision trees or decision forest is a collection of decision trees where an individual decision tree acts as the base classifier. The forest prediction is compiled by taking a vote based on the predictions made by each decision tree [35]. In general, decision forests are generated by inducing different decision trees by perturbing the training data set differently for each decision tree (usually, the training data set is perturbed by excluding some records and/or attributes). As a result of being generated from perturbed data sets, forest trees are compromised in terms of completeness as they partially reflect the training knowledge. Furthermore, forest constituted from decision trees generated for its own purposes closely resembles to a "black box" as the comprehensibility of a single decision tree is lost [8].

Over the last decade or so, a variant of GA called Genetic Programming (GP) has been extensively used for inducing decision trees to overcome different optimization problems such as prediction accuracy and conciseness. The main difference between GA and GP is in the representation of chromosomes. Instead of using a simple data structure like GA, GP uses more complex representation of a chromosome through tree-like structure [13]. However, none of the GP-based techniques proposed so far can effectively address conditional dependencies that may exist among some of the attributes and thus may fall short in generating optimal/near optimal decision tree in terms of prediction accuracy. On the other hand, techniques that adhere to the basic GA philosophy are largely unexplored in constructing decision trees for optimizing the prediction accuracy [8,13]. This inspires us to propose a new, easy-to-implement GA-based decision tree induction technique (NGA-DT) which is more likely to ascertain conditional dependencies among the attributes.

The rest of this paper is organized as follows: In Sect. 2 we provide Background Study that covers a brief introduction to decision tree and some of the major components of GA. In Sect. 3 we discuss some of the well-known and relevant GP-based decision tree induction techniques and their limitations. The proposed GA-based decision tree induction technique is described in Sect. 4. Section 5 discusses the experimental results in detail. Finally, we offer some concluding remarks in Sect. 6.

2 Background Study

2.1 Decision Tree

Hunt's CLS [17] can be credited as the pioneering work for inducing top-down decision trees. According to CLS, the induction of a decision tree starts by selecting a non-class attribute A_i to split a training data set $\boldsymbol{D}$ into a disjoint set of horizontal partitions [30,35]. The purpose of this splitting is to create a purer distribution of class values in the succeeding partitions than the distribution in $\boldsymbol{D}$. The purity of class distribution in succeeding partitions is checked (using an impurity measure) for all contending non-class attributes and the attribute that gives purer class distribution than others is selected as the splitting attribute. The process of selecting the splitting attribute continues recursively in each subsequent partition $\boldsymbol{D}_i$ until either every partition gets the "purest class distribution" or a stopping criterion is satisfied. By "purest class distribution" we mean the presence of a single class value for all records. A stopping criterion can be the minimum number of records that a partition must contain; meaning that if an splitting event creates one or more succeeding partitions with less than the minimum number of records, the splitting is not considered.

Different decision tree induction algorithms that follow the same structure of CLS usually differs in using impurity measures (for measuring the purity of class distribution) in order to find the splitting attributes. For example, C4.5 [30,31] uses Gain Ratio while CART [10] uses Gini Index as impurity measures.

A decision tree consists of nodes (denoted by rectangles) and leaves (denoted by ovals) as shown in Fig. 1. The node of a decision tree symbolizes a splitting event where the splitting attribute (label of the node) partitions a data set according to its domain values. As a result, a disjoint set of horizontal segments of the data set are generated and each segment contains one set of domain values of the splitting attribute. For example, in Fig. 1 "Trouble Remembering" is selected as the splitting attribute in the root node. "Trouble Remembering" has two domain values: "Y" and "N" and thus it splits the data set into two disjoint horizontal segments in such as way that the records of one segment contain "Y" value for "Trouble Remembering" attribute and the records of another segment contain "N" value. The domain values of the splitting attribute designated for the respective horizontal segments are represented by the labels of edges leaving the node.

2.2 Some of the Major Components of GA

Usually, there are five major components in a GA as described in the following.

Initial Population Selection: We already know, a chromosome is a potential solution encoded in a data structure and a set of chromosomes is termed as "Population" in GA. Hence, the "Initial Population" is realized by encoding the first set of chromosomes.

Crossover: Crossover is crucial for the evolution of new population [32,38]. Generally, crossover operation is applied on a chromosome pair (parents) where they swap segments to form the offspring. In this way, a new set of chromosomes representing new solutions are generated [4,20].

Mutation: In Mutation, chromosomes are arbitrarily changed in order to induce more randomness in search directions of the solution space [4,20].

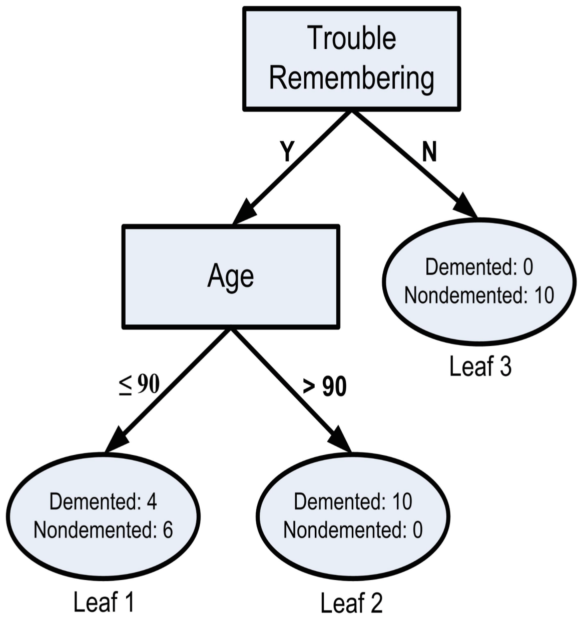

Fig. 1. Decision Tree

Elitist Operation: Elitist operation searches for the best chromosome (the best solution) in a population (from the initial population to a modified population through crossover and mutation operations) [4].

Chromosome Selection for the Next Iteration: After crossover and mutation operations (meaning, after an iteration), the modified population is compared with the immediate previous population (i.e. the population at the beginning of an iteration). If the modified population becomes inferior to the immediate previous population then there is a strong possibility of continuous degradation in subsequent iterations [4]. This may promote the search in wrong directions of the solution space. In order to prevent such scenario, better chromosomes are selected from the modified and the immediate previous populations for the next iteration.

3 Related Works

GP-based decision tree induction techniques that use tree-like chromosomes for encoding decision tree, seem to be the most common in literature [6,8,13,14, 36,41]. Representing a decision tree into a tree-like chromosome gives more flexibility in mimicking rules of a decision tree than a simple data structure such as a string or array. In [6], the authors applied a tree-like chromosome encoding scheme for competitive co-evolution of decision trees. The scheme encodes a binary decision tree by translating the nodes and leaves as 4-tuple:

$node = \{i, N, O, V\}$ where i represents the ID of the attribute tested, N indicates whether it is a node or a leaf and O is the operator used (meaningful for nodes only). V can represent dual values, for nodes it contains the test value and for leaves it contains the binary classification value. Each of the components of the 4-tuple contains numeric value and are subject to modification in the evolution process.

In [33], the authors represented a binary decision tree using a list of 5-tuple: $node = \{t, n, L, R, C\}$. Here, t represents the node number ($t = 0$ at the root), n is the attribute number (meaningful for only nodes), L and R points to left and right children respectively (meaningful for only leaves) and C represents set for counters facilitating the cut point to traverse to left/right child. Similarly, in [36] a decision tree was be represented as 7-tuple in a tree-like chromosome: $node = \{t, label, P, L, R, C, size\}$ where t represents node number ($t = 0$ means the root), $label$ is the class label of a leaf (meaningful for only leaves), P is the pointer to the parent node, L and R represent pointers to left and right children respectively (meaningful for only nodes, for leaves both pointers are $NULL$), C represents a set of registers where for example $C[0]$ stores the ID of the attribute tested and $C[1]$ stores the splitting value for the attribute. Finally, $size$ stores the number of nodes and leaves beneath; such like the size of the root is the size of the whole decision tree while the size of a leaf is 1 [8,36].

In [14], each decision tree of the initial population was generated from a different subset of the training data set in order to stress the decision trees to be as different as possible. All decision trees of the initial population are represented using a list of 4-tuple. This representation allows any node of a decision tree to become the root of a subtree commencing from the node. After initial population selection, two parent decision trees are selected according to roulette wheel technique [25,32] for crossover operation. In crossover, one node is randomly selected from each parent and then subtrees rooted from those nodes are exchanged (see Fig. 2).

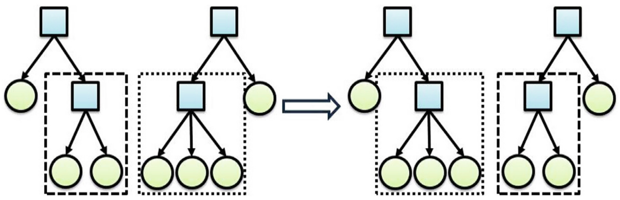

Fig. 2. Crossover

After crossover, mutation operation is applied which involves the exchange of subtrees within a decision tree. In doing so, two nodes within a decision tree are randomly selected and subtrees rooted from those nodes are exchanged (see Fig. 3).

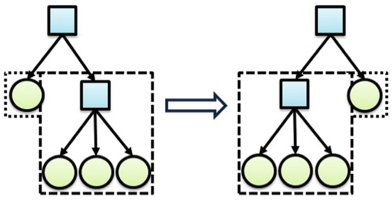

Fig. 3. Mutation

After crossover and mutation operations, it is possible that some decision trees may have logically inconsistent rules. Though, those rules do not affect prediction accuracy (because no records will satisfy logically inconsistent rules), yet the authors [14] opted to prune them. Finally, elitist operation finds the best chromosome (the best decision tree) according to prediction accuracy from the modified population in each iteration.

Despite being flexible in representing decision trees into chromosomes, GP-based techniques have some common limitations. For instance, almost each of the proposed algorithms deals with binary decision tree as it becomes more complicated to represent non-binary decision trees into tree-like chromosomes. The 7-tuple $node = \{t, label, P, L, R, C, size\}$ points to left and right children using two pointers L and R for binary decision tree whereas for non-binary decision trees the number of children may vary in each node. Thus, it becomes more complicated when non-binary decision trees are to be translated into tree-like chromosomes. Evidently, GP is more computationally intensive than GA as GP needs to parse decision trees into complex tree-like chromosomes. Further computational overhead and complications come from applying genetics-inspired operators such as crossover and mutation on those chromosomes. Moreover, with the exchange of subtrees it is difficult to explore conditional dependencies entangled among different sets of attributes in a limited number of iterations.

4 The Proposed GA-Based Decision Tree Induction Technique

The main components of the proposed technique is described as follows.

Chromosome Encoding and Initial Population Selection: The proposed GA-based decision tree induction technique encodes each chromosome (Cr_i) in a one-dimensional array where each cell contains the weight of a non-class attribute. Thus, the length of each chromosome is equal to the number of non-class attributes ($\boldsymbol{m} = \{A_1, A_2, \ldots, A_m\}$) in the training data set. The weights are obtained randomly from a uniform distribution in the interval of $[0, 1]$. As a result, different chromosomes in the initial population obtain different sets of randomly generated weights for the same set of attributes. In the proposed technique, we encode 20 chromosomes to constitute the initial population ($|\mathbf{P}| = 20$) as was done in literature [4] (see Fig. 4).

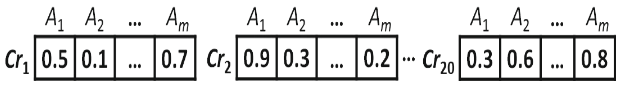

Fig. 4. Initial population in the proposed technique

Elitist Operation: We apply elitist operation to find the chromosome (Cr_b) among the initial population ($\mathbf{P}_{Curr}$) from which the best C4.5 decision tree (in terms of prediction accuracy) is generated. As one C4.5 decision tree is generated from a particular chromosome, the weight distribution remains the same for all nodes of a single C4.5 decision tree, but different weight distributions are likely to be exerted for different C4.5 decision trees. When generating a C4.5 decision tree from a chromosome, at each splitting event, merit values of all non-class attributes are calculated by multiplying the value of impurity measure (such as Gain Ratio [30,31]) of each attribute with the respective random weight which is stored in the chromosome. After the merit values of all non-class attributes are calculated, the attribute with the highest merit value is selected as the splitting attribute.

In this way, the use of random weights introduces a random preference to some attributes in the induction process of a C4.5 decision tree. These "some attributes" ($\boldsymbol{S}$) are likely to be different in different chromosomes and hence gives us the opportunity to test conditional dependencies among different $\boldsymbol{S}$. We know that decision trees are considered to be an unstable classifier as a slight perturbation in a training data set can cause significant differences between decision trees generated from the perturbed and original data sets [3,35]. As a result, a population of chromosomes imposing random weights on attributes causing preferences to different $\boldsymbol{S}$ is more likely to help inducing a number of different decision trees incorporated with some of those conditional dependencies. The best decision tree among them is expected to be the best utilizer of such conditional dependencies. The chromosome generating the best C4.5 decision tree (i.e. the best chromosome) is stored as Cr_b.

Crossover and Mutation: For crossover, we select the best chromosome of $\mathbf{P}_{Curr}$ (Cr_b) as the first chromosome of a pair. To select the second chromosome of the pair, we use roulette wheel technique [25,32] where a chromosome Cr_r ($\neq Cr_b$) is selected with a probability $p(Cr_r) = \frac{PA(Cr_r)}{\sum_{i=1}^{|\mathbf{P}_{Curr}|} PA(Cr_i)}$ ($PA(Cr_r)$ is the prediction accuracy of the decision tree generated from chromosome Cr_r) affirming better chromosomes have greater chance of selection over weaker ones. Once a chromosome pair is selected, they are excluded from the process of choosing the upcoming pairs; all pairs are chosen in the same way as described. The motivation behind roulette wheel selection is to induce some randomness in choosing compatible peer from the best available chromosomes in a population. After pairing the chromosomes, standard 1-point crossover operation is applied on them as described in Fig. 5. A single crossover point is selected randomly between 1 and $|\boldsymbol{m}|$ for each parent pair, where $|\boldsymbol{m}|$ is the number of non-class attributes in the training data set and hence the full length of each chromosome. With the crossover operation, parent chromosomes swap genes (weights); left genes (i.e. genes in the left side of the crossover point) of one chromosome join the right genes of another chromosome. After crossover operation all the parent pairs are converted into offspring pairs with same number of genes.

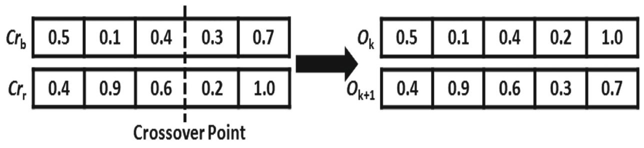

Fig. 5. Crossover in the proposed technique

For mutation, we select one gene randomly from each offspring and then regenerate the weight randomly in the interval of $[0, 1]$. In this way, mutation helps to induce some randomness in search directions. After both crossover and mutation, elitist operation is applied to find the Cr_b.

Chromosome Selection for the Next Iteration: At the end of each iteration, the modified population can be inferior to its immediate previous population. To prevent such degradation, at the end of each iteration we create a pool of chromosomes ($\mathbf{P}_{Pool}$) by adding chromosomes of the modified ($\mathbf{P}_{Mod}$) and its immediate previous population (i.e. the population at the beginning of the iteration, $\mathbf{P}_{Curr}$). Hence, $\mathbf{P}_{Pool}$ consists of 40 chromosomes. We then apply roulette wheel technique to select 20 chromosomes from the 40-chromosome $\mathbf{P}_{Pool}$. The selected chromosomes form the new population ($\mathbf{P}_{Curr}$) for the next iteration. This encourages that good chromosomes representing better solution remain in the process to be given more chance for reproduction. Finally, we obtain the best chromosome (Cr_b) from which we expect to induce an optimal/near optimal C4.5 decision tree.

5 Experimental Results

5.1 Data Set Information and Experimental Setup

We carry out an elaborate experimentation on thirty well known data sets that are publicly available from the UCI Machine Learning Repository [23] covering a variety of areas. The data sets used in the experimentation are described in Table 1. For example, the Car Evaluation data set has six non class attributes, 1728 records distributed in four distinct class values. The data sets are presented in Table 1.

We already know, till date C4.5 [30,31] remains to be one of the most accurate and popular decision tree induction algorithms [18,24] and any improvement beyond C4.5 can render significant influence over its large application domain.

Table 1. Description of the data sets

Data Set name (DS)	Non-class attributes	Records	Distinct class values
Abalone (AB)	08	4177	28
Balance Scale (BS)	04	625	3
Breast Cancer (BC)	33	194	2
Car Evaluation (CE)	06	1728	4
Chess (CHS)	36	3196	2
Credit Approval (CA)	15	653	2
Dermatology (DER)	34	358	6
Glass Identification (GI)	09	214	6
Hayes-Roth (HR)	04	132	3
Hepatitis (HEP)	19	80	2
Image Segmentation (IS)	19	2310	7
Ionosphere (ION)	34	351	2
Iris (IRS)	04	150	3
Letter Recognition (LR)	16	20000	26
Libras Movement (LM)	90	360	15
Liver Disorder (LD)	06	345	2
Nursery (NUR)	08	12960	5
Pen-Based Recognition of Handwritten Digits (PD)	16	10992	10
Pima Indians Diabetes (PID)	08	768	2
Seeds (SDS)	07	210	3
Sonar (SON)	60	208	2
Statlog Heart (SH)	13	270	2
Statlog Vehicle (SV)	18	846	4
Teaching Assistant Evaluation (TAE)	05	151	3
Thyroid Disease (TD)	05	215	3
Tic-Tac-Toe (TTT)	09	958	2
Wine (WNE)	13	178	3
Wine Quality (WQ)	11	6497	7
Yeast (YST)	08	1484	10
Zoo (ZOO)	16	101	7

Hence, we apply the proposed GA-based technique (NGA-DT) in order to induce optimal/near optimal C4.5 decision tree. The main purpose of our experimentation is to demonstrate how much improvement (in terms of prediction accuracy) an optimal/near optimal C4.5 decision tree can offer over a regular C4.5 decision tree. Therefore, we use the same settings for generating both NGA-DT and C4.5 decision trees. NGA-DT uses the same impurity measure (Gain Ratio [30,31]) and the entire training data set (not different subsets/bootstrap samples of the training data set) as used in a regular C4.5 decision tree. The minimum Gain Ratio/merit value is set to 0.01 for any attribute to qualify for splitting a node, Each leaf node of a tree contains at least two records and no further post-pruning is applied. In [14], decision trees are generated from different subsets of the training data set and thus compromised in terms of completeness as they partially reflect the training knowledge. Furthermore, those decision trees are not fully grown as logically inconsistent rules are pruned from them. Therefore, a decision tree generated from [14] cannot be regarded as a variant of a regular C4.5 decision tree and hence we exclude [14] from the comparison spectrum.

The experimentation is conducted by a machine with Intel(R) 3.4 GHz processor and 8 GB Main Memory (RAM) running under 64-bit Windows 7 Enterprise Operating System. All the results reported in this paper are obtained using 10-fold-cross-validation (10-CV) [7,21,22] for every data set. In 10-CV, a data set is divided randomly into 10 segments and from the 10 segments each time one segment is regarded as the test data set (out of bag samples) and the rest 9 segments are used for training decision trees. In this way, 10 training and 10 corresponding testing segments are generated. In our experimentation, we generate 20 decision trees from each population and hence a total of 20×20 (20 iterations) $= 400$ decision trees from each training segment and then evaluate their performance on the training and the corresponding testing segments. The best results reported in this paper are stressed through **bold-face**.

5.2 Comparison Between C4.5 and NGA-DT

Prediction Accuracy (PA) is one of the most important performance indicators for any decision tree algorithm [2,5]. In Table 2 we present the PA (in percentage) of C4.5 and NGA-DT on training and testing segments of all data sets considered. From Table 2 we see that NGA-DT performs better than C4.5 on training segments of all thirty data sets. This implies that training segments are more correctly reflected through NGA-DT. Hence, NGA-DT can be more reliable for knowledge discovery compared to C4.5.

It is shown in literature that maximizing the PA on training data set may lead to improving the generalization performance [34]. The results presented in Table 2 validate the proposition as NGA-DT performs better than C4.5 on testing segments of twenty three data sets and for one data set they have a draw. Now, to access the significance of improvement on the testing segments, we conduct a statistical significance analysis using Wilcoxon Signed-Ranks Test [1,39]. Wilcoxon Signed-Ranks Test is said to be more preferable to counting only significant wins and losses for comparison between two classifiers over multiple data sets [12]. We observe that PAs do not follow a normal distribution and thus do not satisfy the conditions for parametric tests. Hence, we perform a non-parametric one-tailed Wilcoxon Signed-Ranks Test [1,39] for $n = 30$ (number of data sets used) with the significance level: $\alpha = 0.005$. Thus, the critical value is

Table 2. Prediction accuracies

DS	C4.5 on training segment	NGA-DT on training segment	C4.5 on testing segment	NGA-DT on testing segment
AB	25.26	**31.95**	18.10	**21.83**
BS	81.57	**82.05**	**65.60**	64.43
BC	81.45	**84.32**	70.55	**72.65**
CE	96.50	**96.60**	**94.10**	94.09
CHS	96.02	**99.78**	95.97	**99.25**
CA	86.67	**87.80**	86.37	**86.90**
DER	71.81	**94.54**	71.87	**93.87**
GI	83.55	**90.70**	65.85	**69.90**
HR	66.84	**70.04**	**46.97**	**46.97**
HEP	94.44	**98.61**	82.50	**87.50**
IS	95.40	**98.56**	94.46	**95.37**
ION	96.30	**97.91**	**92.02**	90.02
IRS	97.48	**98.45**	**95.33**	94.00
LR	75.92	**79.29**	71.05	**73.98**
LM	90.49	**91.98**	64.72	**65.28**
LD	76.03	**77.42**	**67.29**	64.83
NUR	98.81	**99.18**	97.00	**97.06**
PD	97.83	**98.77**	95.03	**95.59**
PID	79.89	**82.12**	72.57	**73.21**
SDS	95.50	**98.62**	91.43	**92.86**
SON	93.33	**98.40**	**72.21**	72.14
SH	90.33	**93.33**	75.55	**78.15**
SV	75.59	**85.99**	65.96	**71.08**
TAE	65.93	**73.29**	47.58	**51.63**
TD	97.73	**99.27**	92.84	**93.90**
TTT	91.11	**92.25**	82.59	**82.60**
WNE	98.88	**99.81**	93.53	**95.67**
WQ	50.55	**59.33**	45.17	**50.33**
YST	60.75	**68.04**	49.39	**55.24**
ZOO	97.14	**97.36**	89.00	**90.00**
Avg.	83.64	***87.52***	75.09	***77.34***

calculated to be: 109 [26,37]. The test statistic for the Wilcoxon Signed-Ranks Test based on the PAs of NGA-DT and C4.5 (on the testing segments) is calculated to be: 70. As the test statistic remains lower than the critical value, we understand that NGA-DT significantly improves the generalization performance of C4.5.

6 Conclusion

In this paper, we propose a new, less complicated GA-based decision tree induction technique called NGA-DT which is more likely to ascertain conditional dependencies among the attributes. From the experimental results, we see that NGA-DT significantly improves the generalization performance of C4.5. It is also shown that NGA-DT can be more reliable for knowledge discovery compared to C4.5. The structure of NGA-DT is developed in such a way that other renowned decision tree induction algorithms (such as CART [10]) can be used instead of C4.5 without any modification.

A major drawback of NGA-DT (which involves many other GA-based techniques) is that it has a large memory and computational overhead. The memory and computational overhead of NGA-DT is roughly 1200 times greater than that of C4.5. Hence, NGA-DT may not be suitable for time-critical applications where quick results are appreciated. However, when additional time, memory and processing power is available, NGA-DT can utilize their full potential. Furthermore, the structure of NGA-DT can be implemented in a parallel environment in order to address higher computational time.

References

1. Abellan, J.: Ensembles of decision trees based on imprecise probabilities and uncertainty measures. Inf. Fusion **14**, 423–430 (2013)
2. Adnan, M.N., Islam, M.Z.: ComboSplit: combining various splitting criteria for building a single decision tree. In: Proceedings of the International Conference on Artificial Intelligence and Pattern Recognition, pp. 1–8 (2014)
3. Adnan, M.N., Islam, M.Z.: Forest CERN: a new decision forest building technique. In: Proceedings of the 20th Pacific Asia Conference on Knowledge Discovery and Data Mining (PAKDD), pp. 304–315 (2016)
4. Adnan, M.N., Islam, M.Z.: Optimizing the number of trees in a decision forest to discover a subforest with high ensemble accuracy using a genetic algorithm. Knowl.-Based Syst. **110**, 86–97 (2016)
5. Adnan, M.N., Islam, M.Z., Kwan, P.W.H.: Extended space decision tree. In: Wang, X., Pedrycz, W., Chan, P., He, Q. (eds.) ICMLC 2014. CCIS, vol. 481, pp. 219–230. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45652-1_23
6. Aitkenhead, M.J.: A co-evolving decision tree classification method. Expert Syst. Appl. **34**(1), 18–25 (2008)
7. Arlot, S.: A survey of cross-validation procedures for model selection. Stat. Surv. **4**, 40–79 (2010)
8. Barros, R.C., Basgalupp, M.P., de Carvalho, A.C.P.L.F., Freitas, A.A.: A survey of evolutionary algorithm for decision tree induction. IEEE Trans. Syst. Man Cybern. - Part C: Appl. Rev. **42**(3), 291–312 (2012)
9. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, New York (2008)
10. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth International Group, Belmont (1985)
11. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Discov. **2**, 121–167 (1998)

12. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
13. Espejo, P.G., Sebastian, S., Herrera, F.: A survey on the application of genetic programming to classification. IEEE Trans. Syst. Man Cybern. - Part C: Appl. Rev. **40**(2), 121–144 (2010)
14. Fu, Z., Golden, B., Lele, S., Raghavan, S., Wasli, E.: Genetically engineered decision trees: population diversity produces smarter trees. Oper. Res. **51**(6), 894–907 (2003)
15. Han, J., Kamber, M.: Data Mining Concepts and Techniques. Morgan Kaufmann Publishers, San Francisco (2006)
16. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. MIT Press, Cambridge (1992)
17. Hunt, E., Marin, J., Stone, P.: Experiments in Induction. Academic Press, New York (1966)
18. Kamber, M., Winstone, L., Gong, W., Cheng, S., Han, J.: Generalization and decision tree induction: efficient classification in data mining. In: Proceedings of the International Workshop Research Issues on Data Engineering, pp. 111–120 (1997)
19. Kataria, A., Singh, M.D.: A review of data classification using k-nearest neighbour algorithm. Int. J. Emerg. Technol. Adv. Eng. **3**(6), 354–360 (2013)
20. Kim, Y.W., Oh, I.S.: Classifier ensemble selection using hybrid genetic algorithms. Pattern Recogn. Lett. **29**, 796–802 (2008)
21. Kurgan, L.A., Cios, K.J.: Caim discretization algorithm. IEEE Trans. Knowl. Data Eng. **16**, 145–153 (2004)
22. Li, J., Liu, H.: Ensembles of cascading trees. In: Proceedings of the Third IEEE International Conference on Data Mining, pp. 585–588 (2003)
23. Lichman, M.: UCI machine learning repository. http://archive.ics.uci.edu/ml/datasets.html. Accessed 15 Mar 2016
24. Lim, T.S., Loh, W.Y., Shih, Y.S.: A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. Mach. Learn. **40**, 203–229 (2000)
25. Liu, Y., Shen, Y., Wu, X.: Automatic clustering using genetic algorithms. Appl. Math. Comput. **218**, 1267–1279 (2011)
26. Mason, R., Lind, D., Marchal, W.: Statistics: An Introduction. Brooks/Cole Publishing Company, New York (1998)
27. Murthy, S.K.: On growing better decision trees from data. Ph.D. thesis, The Johns Hopkins University, Baltimore, Maryland (1997)
28. Murthy, S.K.: Automatic construction of decision trees from data: a multidisciplinary survey. Data Min. Knowl. Discov. **2**, 345–389 (1998)
29. Murthy, S.K., Kasif, S., Salzberg, S.S.: A system for induction of oblique decision trees. J. Artif. Intell. Res. **2**, 1–32 (1994)
30. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo (1993)
31. Quinlan, J.R.: Improved use of continuous attributes in C4.5. J. Artif. Intell. Res. **4**, 77–90 (1996)
32. Rahman, M.A., Islam, M.Z.: A hybrid clustering technique combining a novel genetic algorithm with k-means. Knowl.-Based Syst. **71**, 345–365 (2014)

33. Shirasaka, M., Zhao, Q., Hammami, O., Kuroda, K., Saito, K.: Automatic design of binary decision trees based on genetic programming. In: Second Asia-Pacific Conference on Simulated Evolution and Learning. Australian Defense Force Academy, Canberra (1998)
34. Tamon, C., Xiang, J.: On the boosting pruning problem. In: López de Mántaras, R., Plaza, E. (eds.) ECML 2000. LNCS (LNAI), vol. 1810, pp. 404–412. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45164-1_41
35. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Pearson Education, London (2006)
36. Tanigawa, T., Zhao, Q.: A study on efficient generation of decision trees using genetic programming. In: Genetic and Evolutionary Computation Conference (GECCO'2000), pp. 1047–1052. Morgan Kaufmann (2000)
37. Triola, M.F.: Elementary Statistics. Addison Wesley Longman Inc., Reading (2001)
38. Whitley, D.: A genetic algorithm tutorial. Stat. Comput. **4**, 65–85 (1994)
39. Wilcoxon, F.: Individual comparison by ranking methods. Biometrics **1**, 80–83 (1945)
40. Zhang, G.P.: Neural networks for classification: a survey. IEEE Trans. Syst. Man Cybern. **30**, 451–462 (2000)
41. Zhao, H.: A multi-objective genetic programming programming approach to developing pareto optimal decision trees. Decis. Support Syst. **43**(3), 809–826 (2007)