A Hierarchical Nonlinear Discriminant Classifier Trained Through an Evolutionary Algorithm

Ziauddin Ursani[1,2(✉)] and David W. Corne[1,2]

[1] Heriot Watt University, Edinburgh, UK
ziaursani@yahoo.com
[2] Route Monkey Limited, Livingston, UK

Abstract. This work builds on our earlier two papers where we developed method to train nonlinear discriminant classifier for 4-feature datasets. In this paper, the method has been formalized to include any number of features. A hierarchical nonlinear discriminant classifier builds models using a constrained pattern of feature combinations. The model is far more expressive than naïve Bayes, for example, which does not consider feature combinations at all; and the model is far more parsimonious and scalable than unconstrained genetic programming (for example), which does not rule out any feature combinations. The method can be used for knowledge acquisition and decision-making expert system as it can retrieve 100% accurate model from the dataset. The method can also be used for classification of unseen data. The method has been tested on popular test datasets present in the UCI repository. Two approaches are presented to apply a learned model to the test set. The first method consists of application of a single exact hierarchical model on the test set; another method is the application of a weighted sum of models present in each hierarchy. Results of this approach on the datasets studied here are found to be very competitive with the results in recent literature.

Keywords: Hierarchical model · Weighted sum model · Nonlinear model Supervised learning

1 Introduction

Helping computers to learn classification task is the very important area of machine learning, which has now dominated the artificial intelligence literature since last several decades. Supervised Learning has remained subject of research throughout this period. To evaluate the performance of the supervised learning methodology, the dataset is divided into training set and the test set. The model is trained on the training set and then is applied on the test set. Though it is not guaranty that the more accurate model on the training set produces more accurate results on the test set but retrieving accurate models from the datasets has always been subject of interest for knowledge acquisition and decision-making problems [1, 2]. The most popular system in this area is C4.5 inductive decision tree learners [3]. In this paper, we propose a method which is capable of highly accurate low-complexity models of the training set (i.e. models have

Y. Tabii et al. (Eds.): BDCA 2018, CCIS 872, pp. 273–288, 2018.
https://doi.org/10.1007/978-3-319-96292-4_22

very few parameters, and achieved 100% accuracy in training on the cases studied here). High accuracy on a training set provides, of course, generally no information about generalization quality in machine learning; however, when such accuracy is regularly obtained with a low complexity model, it becomes of potential interest for fields such as knowledge acquisition, especially if the model also performs well on test sets. The method is extension of our preliminary work [4, 5], now rendered more scalable for datasets with many features. The model can be described as a hierarchical nonlinear discriminant classifier that exploits a constrained pattern of feature combinations in a fixed tree data structure. The model is far more expressive than, for example, naïve Bayes [6], which does not consider feature combinations at all; and the model is far more parsimonious and scalable than unconstrained genetic programming [7], which does not rule out any feature combinations. The nonlinear discriminant classifiers have been in the literature now for a considerable time, such as Kernel based nonlinear discriminant classifiers [8, 9]. However, in this paper hierarchical nonlinear discriminant classifier model is proposed, which is constructed automatically through randomized training procedure. The model is stochastic, trained via an evolutionary algorithm, therefore produces potentially different models in each run. However, it seems to reliably find 100% accurate models of the training set in the cases we have studied so far, and present herein. This paper contains some examples of these models produced on three datasets Iris Flower, Balance Scale and Car Evaluation. These datasets are popular test cases for classification and knowledge acquisition problems and are present in the UCI machine learning repository [10]. Later, the method is used for classification of unseen data on the same datasets. The model is trained on the training set which is a randomly chosen subset of the original set. The trained model is then applied to classify the test set, which is a subset of the original set complimentary to the training set. Therefore, data in the test set is not seen by the model during training. The paper proposes two methods for application of model on the test set. One method is to exactly apply same hierarchical model on the test set and another method is to produce a model that is weighted sum of models present in each hierarchy of the trained model. The results are competitive with the state of art literature.

The rest of the paper is structured as follows. In Sect. 2, a tree generation model for the feature set of any size is proposed. Section 3 consists of description of evolutionary algorithm that trains the tree data structure model. The detailed description of the three test datasets is given in the Sect. 4. Experimental design of supervised learning for classification of these datasets and their results are discussed in Sect. 5. Section 6 concludes the findings and speculates on the future work. Finally, an appendix is given which gives some examples of accurate models trained through an evolutionary algorithm on the complete test datasets.

2 Tree Generation Model

A tree generation model generates a tree that can represent a mathematical model consisting of full feature set of the dataset regardless of its size. The total number of nodes in the tree generation model is governed by Eq. 1.

$$n = 3 * f - 1 \tag{1}$$

where

n = number of nodes in the tree
f = number of features in the dataset

The tree essentially consists of three types of nodes.

2.1 Weight Nodes n_w

These are the tail nodes of the tree which contain the weight of the features; therefore, number of weight nodes is equal to number of features. The id numbers of these weight nodes start from $2 \times f$ and end at $3 \times f - 1$. The value of weight ranges between 0–1. All the weight nodes are present at the last level of the tree or they are leaf nodes.

2.2 Feature Nodes n_f

These are the nodes preceding to weight nodes. The feature nodes contain the actual feature values and hence are also equal to number of features. The id numbers of these nodes start from f and end at $2f - 1$. All the feature nodes are present at the second last level of the tree or one level before the leaf nodes.

2.3 Operator Nodes n_o

The nodes preceding to the feature nodes are operator nodes. These nodes contain the information about mathematical operator, which is supposed to be applied on the two expressions represented by two branches emanating from this node. The number of the operator nodes are one less than the feature nodes i.e., $n_o = n_f - 1$. The id numbers of operator nodes start from 1 and end at $f - 1$. The operator nodes are present at different hierarchy levels of the tree starting from the first level to the third last level. At the third last level, the operator nodes follow the following rule.

$$n_o^3 = INT\left(\frac{f}{2}\right) \tag{2}$$

where

n_o^3 = Number of operator nodes at the third last level

On the levels preceding to 3rd last level, the operator nodes follow following rule.

$$n_o^m = \begin{cases} INT\left(\frac{n_o^{m-1}}{2}\right) + 1, & if\ n_o^{m-1},\ n_x^{m-2}\ are\ odd \\ INT\left(\frac{n_o^{m-1}}{2}\right), & otherwise \end{cases} \tag{3}$$

where

n_o^m = number of operator nodes at m^{th} last level
n_x = can be any nodes i.e. feature nodes or operator nodes depending on the level of tree.

The Eq. 3 says that the number of operator nodes at any level of tree depends on the number of nodes at two succeeding levels. If the number of nodes at two succeeding levels are odd then the number of operator nodes will be one more than the number of nodes in the other case. The operator nodes contain integer value from 1–4, each representing each of four mathematical operators $+, -, \times, \div$ respectively.

Let us explain above model with the car evaluation dataset which has six features. The tree in Fig. 1 is representative of this model. The tree in Fig. 1 contains 17 nodes which follows the Eq. 1. The nodes 12–17 are weight nodes. The nodes 6–11 are feature nodes and the nodes 1–5 are operator nodes. At the third last level, there are three operator nodes 3–5, which follow the Eq. 2. At fourth last level (2nd level), there is only node 2 and at the fifth last level (1st level) again there is only node 1. The nodes at first and second level follow the Eq. 3. The node at 1st level follows conditional part of Eq. 3 and node at 2nd level follows otherwise part of Eq. 3.

Now if the weight nodes 12–17 contain weight values w_1–w_6 respectively, the feature nodes 6–11 contain values f_1–f_6 respectively, the operator nodes 4–5 contain the value +, the operator nodes 1–3 contain the values $\div, \times, -$ respectively then the phenotype equivalent $\in$ of this tree structure is given in Eq. 4.

$$\in = \frac{w_1 f_1 - w_2 f_2}{(w_3 f_3 + w_4 f_4) \times (w_5 f_5 + w_6 f_6)} \tag{4}$$

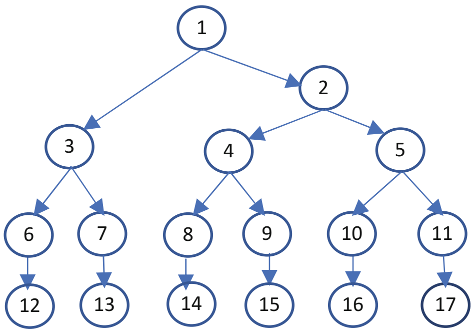

Fig. 1. A tree model for six feature set

It can be seen from the example Eq. 4, that values at weight nodes are multiplied with corresponding feature nodes and then resultant expressions are subjected to operators represented in the operator nodes.

3 Evolutionary Algorithm (EA)

The evolutionary algorithm trains the tree data structure explained in Sect. 2. The evolutionary algorithm can be applied on the whole dataset for knowledge acquisition or it can also be applied on the randomly chosen part of the dataset for training purpose. The algorithm follows the hierarchical procedure used in our earlier work [5], summarized here as follows.

a. Create two lists i.e. sample list and model hierarchy list
b. Store the number of samples under examination in the sample list.
c. Initialize model hierarchy list with level $n = 0$.
d. The EA starts with the random generation of population of solutions. Each solution consists of tree data structure presented in section 2.
e. The EA evaluates each solution of the population according to fitness and unfitness function, discussed in detail later in this section.
f. The individuals for reproduction of next generation are selected using binary tournament selection.
g. The next generation is produced through reproductive procedures consisting of crossover and mutation operators as described in [5].
h. If termination condition is false then go to step e.
i. Set $n = n + 1$ and store the trained model in the model hierarchy list
j. Delete the classified samples from the sample list.
k. If sample list is still non-empty then go to step d.
l. Terminate the program.

Procedure 1: Hierarchical application of evolutionary algorithm

It can be seen from the above procedure, that the algorithm continues to train models iteratively until all samples are classified. Finally, all the models can be put in a hierarchical way to represent decision model of the whole dataset or the training set under examination. The interesting thing about these models is that every single model correctly classifies some of the samples but it doesn't misclassify any of the samples. This is accomplished through combination of fitness and unfitness function (step e). The primary objective is to maximize fitness function and secondary objective is to minimize unfitness function. The fitness function is equal to number of classified samples. Unfitness function is the value of partition wall that is incorporated into the model to prevent model from misclassifying the samples. The model is probabilistic. It measures probability of sample to be member of each class. These probabilities are computed with the help of the probabilistic model based on the distance of phenotype value $\in$ (example Eq. 4) of sample i from the phenotype mean of class j. Figure 2 along with Eq. 5 illustrates this probabilistic principle of membership.

$$p_i^j = \frac{\delta}{\mu} \tag{5}$$

where

p_i^j = probability that sample i is member of class j
δ = Distance of phenotype value for sample i from estimated maximum/minimum of class j
μ = Distance of estimated mean of phenotype value of all samples of class j in the training set from estimated minimum/maximum of the values of members of class j.

It is clear from the Fig. 2 and Eq. 5, that the sample will have greater probability of class membership when it is closer to the mean position of the class. Its probability of class membership decreases when it goes farther and becomes negative when it goes farther than the estimated minimum/maximum of the class phenotype value. We know that in standard probability theory probability ranges between (0–1), however, according to Eq. 5, its value can go much below zero and we keep it as it is because it is useful in our class membership function later discussed in Eq. 9. The estimated mean of phenotype value of class j is calculated from the training set as follows.

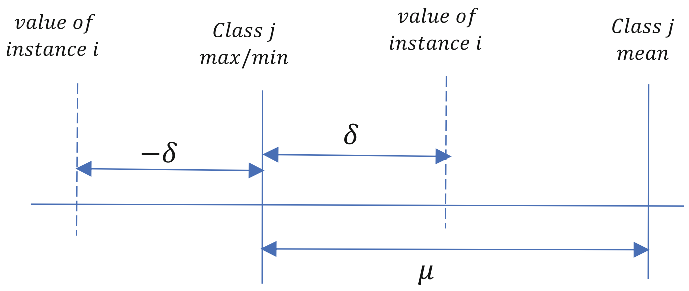

Fig. 2. A principle of probabilistic membership

$$\epsilon_{mean}^{j} = \frac{\sum_{i=1}^{i=t_j} \in_i}{t_j + \Delta} \tag{6}$$

where

$\in_i$ = Phenotype value for sample i according to evaluation of model described in Sect. 2 (for example, Eq. 4)
t_j = Number of samples in the training set of class j
Δ = Predictive parameter for larger sample = 1.0

The estimated maximum and minimum of phenotype value of class j are modelled as follows.

$$\epsilon_{\substack{max \\ min}}^{j} = \epsilon_{mean}^{j} \pm 3.0 * \epsilon_{sd}^{j} \tag{7}$$

where

ϵ^j_{mean} = estimated mean of set of phenotype values of member samples of class j in the training set

$\epsilon^j_{\substack{max\\min}}$ = estimated maximum/minimum of set of phenotype values of member samples of class j in the training set

ϵ^j_{sd} = estimated standard deviation of set of phenotype values of member samples of class j in the training set

The estimated standard deviation of phenotype value of class j is modelled as follows.

$$\epsilon^j_{sd} = \frac{\sum_{i=1}^{i=t_j}\left(\epsilon^j_i - \epsilon^j_{mean}\right)^2}{t_j - \Delta} \tag{8}$$

The task of classifying the sample i is achieved through class membership function as given below.

$$\emptyset_i = k \text{ iff } \; P^k_i > \forall^{j\neq k}_{j=1,\mathrm{n}_c}\left(P^j_i + \nabla\right) \tag{9}$$

where

$\emptyset_i$ = class of sample i
∇ = Safety partition to avoid misclassification during training (unfitness function)
n_c = Total number of classes in the dataset

It is clear from the Eq. 9, that the sample is classified into the class with which it has highest probability of class membership among all the classes. However, it remains unclassified if highest probability of class membership is not greater enough than the second highest probability of class membership to overcome the obstacle of unfitness function ∇. Following are the steps of evaluation procedure of chromosome.

a. The evaluation starts by setting unfitness function/safety partition $\nabla = 0$.
b. Set sample number $i = 0$
c. Increment $i = i + 1$
d. Apply model in relation 9 to classify the sample i
e. If the model misclassifies a sample i to a wrong class k, then $\nabla = P^k_i - \max\left(\forall^{j\neq k}_{j=1,\mathrm{n}_c} P^j_i\right)$ go to step b.
f. Terminate the procedure

Procedure 2: Evaluation procedure of chromosome

It is clear from the step e that the value of unfitness function is raised to minimum threshold level to avoid misclassification. Due to this raised value of safety partition none of the probability values of any class membership satisfy the condition placed in model (relation 9). Therefore, the sample i remains unclassified. Since the model has now been modified, therefore procedure of evaluation starts again from first sample with the new value of unfitness function. The procedure continues until all samples of training dataset are examined under same value of unfitness function and none of the samples are misclassified. Now the chromosome fitness value is composite of its fitness and unfitness function. The fitness function is number of classified samples and value of ∇ is unfitness function.

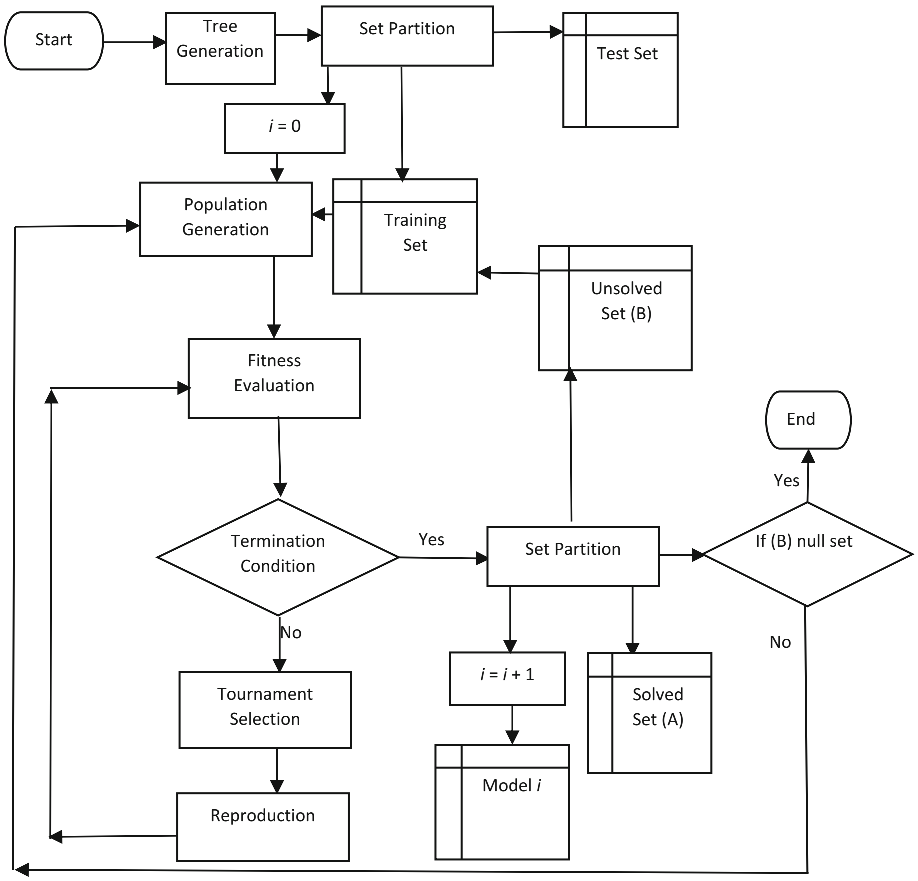

Fig. 3. Flowchart of hierarchical model evolution

Now the primary objective is to maximize fitness function i.e., maximize number of classified samples and the secondary objective is to minimize unfitness function i.e. value of ∇. Therefore, while comparing fitness of chromosomes, a chromosome with higher number of classified samples is considered better regardless of value of its unfitness function. The value of unfitness function is only considered when the two chromosomes have equal score in number of classified samples. The flowchart of whole procedure is depicted in Fig. 3.

The flowchart starts from the process of tree generation, which is described in Sect. 2. Understandably the tree is generated after reading the dataset. The dataset is then partitioned into training set and test set through the procedure of set partition, which is entirely random procedure but it ascertains the proportional representation of each class in the training set. The variable *i* is initialized with zero. This variable represents a model or hierarchy number, which will be incremented later with the generation of model. The evolutionary algorithm starts with generation of random population followed by typical cycle of evolutionary iteration consisting of fitness evaluation, selection and reproduction until termination condition is achieved. After the termination the training set is further partitioned into the solved set and unsolved set. The solved set consists of classified samples of the training set by the trained model while the unsolved set consists of samples which the model failed to classify. Now if this unsolved set is not a null set, then it is considered as a training set for the next phase of application of evolutionary algorithm, which is again starts from generation of random population of solutions. This iterative procedure continues until unsolved set becomes null set.

4 Description of Datasets

This paper considers three datasets which are taken from UCI repository [10] to analyze the performance of the method proposed. Following is the description of those datasets.

4.1 Iris Flower

This is a botanical dataset. The dataset contains 150 samples of 3 species of iris flower called Setosa, Virginica and Versicolour. The dataset has 50 samples of each class, with details of four features i.e., sepal width, sepal length, petal width and petal length. The dataset was created by Anderson in 1935 [11] and later was popularized by Sir Fisher in 1936 [12]. The dataset is most popular in pattern recognition and classification domain.

4.2 Balance Scale

This is a psychological dataset. This dataset was created by Siegler in 1976 [13] to model psychological experimental results. The dataset contains 625 examples of persons with attributes left weight, right weight, left distance and right distance. This data is helpful in determining whether person is balanced, right tipped or left tipped. The dataset contains 288 examples for each right and left tipped people while only 49 examples for the balanced people. This is a very popular test case in the classification domain.

4.3 Car Evaluation

This is a decision-making dataset. The dataset has six attributes namely buying price, maintenance cost, number of doors, maximum number of accommodable persons, size of lug-boot and level of safety measures. The dataset has total of 1728 samples. The dataset has four classes unacceptable, acceptable, good and very good. The unacceptable class has 1210 samples. The acceptable class has 384 samples. The good class has 69 samples and very good class has 65 samples.

Table 1 summarizes feature list of these datasets. Column 1 contains name of the dataset, column 2 provides number of features in the dataset and columns 3–8 provide name of the feature corresponding to label used in the column head. These labels are later used to represent classifier models of the dataset in Table 5 in the Appendix. Table 2 summarizes the class list of the dataset. Again column 1 contains name of the dataset, column 2 informs about number of classes in the dataset and columns 3–6 give name of the class corresponding to label used in the column head. These labels are later used in Table 6 in Appendix to give statistical data about generated models.

Table 1. Feature description for each dataset

Dataset	Number of features	f_1	f_2	f_3	f_4	f_5	f_6
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Balance scale	4	Left weight	Left distance	Right weight	Right distance	-	-
Iris flower	4	Sepal length	Sepal width	Petal length	Petal width	-	-
Car evaluation	6	Buying cost	Maintenance cost	Number of doors	Number of seats	Size of lug-boot	Level of safety

Table 2. Class description for each dataset

Dataset	Number of classes	c_1	c_2	c_3	c_4
(1)	(2)	(3)	(4)	(5)	(6)
Balance scale	3	Balanced	Left tipped	Right tipped	
Iris flower	3	Setosa	Virginica	Versicolour	
Car evaluation	4	Unacceptable	Acceptable	Good	Very good

5 Experimental Design and Analysis

The experiments are performed on the datasets described in Sect. 4. The objective of experiments is two-fold. First objective is to retrieve 100% accurate models from complete datasets. The results of these experiments are included in the Appendix. The second objective is to develop models on the randomly generated training sets and then verify those models on the test set. On the test set trained models are applied in two different ways. First method is to apply models in hierarchical way as stored in model hierarchy list described in procedure 1. Following is the stepwise method for hierarchical application of models.

a. Create a sample list
b. Store the samples to be tested in the sample list.
c. Set pointer in the model hierarchy list with level $n = 1$.
d. Apply the pointed model on the sample list for classification (step explained further in appendix in procedure 4)
e. Delete the classified samples from the sample list.
f. If sample list is non-empty then increment the pointer in the model hierarchy list $n = n + 1$ and go to step d.
g. Terminate the program.

Procedure 3: Evaluation Procedure of the test set

The second method is weighted sum method, i.e., weighted sum of all models present in the model hierarchy list is produced and applied on the test dataset. The weights to the models are assigned based on the fitness i.e. number of classified samples by the model. The experiments are performed on randomly generated training sets of three different sizes equivalent to around 50%, 80% and 90% of the original size of the dataset. The training sets are generated in a way that samples of each class are chosen proportionally for proper representation of each class in the training set. 30 simulations are run on each dataset. However, each simulation has different randomly generated training set. The results of experiments are summarized in Table 3.

Table 3. Classification results on the datasets.

Dataset	Classification method	Size of training set	Best results	Average results	% age of accurate results
(1)	(2)	(3)	(4)	(5)	(6)
Iris flower	Weighted sum	50%	98.67%	94.31%	0.00%
		80%	100.00%	95.44%	23.33%
		90%	100.00%	96.44%	60.00%
	Hierarchical	50%	98.67%	93.51%	0.00%
		80%	100.00%	94.22%	6.67%
		90%	100.00%	94.44%	36.67%
Balance scale	Weighted sum	50%	100.00%	97.44%	3.33%
		80%	100.00%	98.53%	43.33%
		90%	100.00%	96.13%	40.00%
	Hierarchical	50%	100.00%	99.05%	23.33%
		80%	100.00%	99.76%	80.00%
		90%	100.00%	99.41%	73.33%
Car evaluation	Weighted sum	50%	85.19%	78.96%	0.00%
		80%	88.12%	79.78%	0.00%
		90%	86.05%	80.78%	0.00%
	Hierarchical	50%	95.14%	93.29%	0.00%
		80%	97.68%	94.88%	0.00%
		90%	98.84%	95.81%	0.00%

In Table 3, column-1 contains name of the dataset, column 2 provides name of the classification method, column 3 gives size of the training set in terms of percentage to original size, columns 4–5 informs about best and average result of 30 simulations respectively. The results are in terms of percentage of correctly classified samples. Finally, column 6 provides percentage of accurate results i.e., the percentage of number of simulations out of 30 simulations where 100% accurate results are obtained.

It can be seen from the Table 3, that better results are obtained with weighted sum method on the iris flower dataset while on the balance scale and car evaluation datasets better results are obtained with hierarchical method of classification. It can also be noticed that best results are obtained on training set of 80% size on balance scale and car evaluation dataset, while for iris flower dataset best results are obtained at the training set of 90% size. The hierarchical method has been most successful on the balance scale dataset with up to 99.76% average results on test set with the training set of 80% size, whereas accurate results on test set have been obtained on 80% of simulations.

Table 4, has also been prepared to compare results with other methods. In Table 4, column 1 gives reference of the method, column 2 provides name of the dataset, column 3 informs about number of simulations/cross validation, size of training set in terms of percentage to original dataset is given in column 4. Columns 5–6 give average and best results respectively of all the simulations in terms of percentage of correctly classified samples.

Table 4. Classification results in the literature

Ref.	Dataset	Sim/X-validation	Size of Tr. set	Average results	Best results
(1)	(2)	(3)	(4)	(5)	(6)
[12]	Iris	-	80%	-	100%
	Balance		50%	-	88.14%
[13]	Iris	10	90%	94.67%	-
	Balance		90%	78.88%	-
[14]	Iris	10-fold X-validation	90%	94.0%	94.0%
	Balance		90%	89.1%	89.1%
[15]	Iris	15x10-fold X-validation	90%	95.65%	-
	Balance		90%	85.28%	-
	Car		90%	98.48%	-
[16]	Iris	10-fold X-validation	90%	87.22%	-
	Balance		90%	71.27%	-
	Car		90%	70.12%	-
[17]	Iris	10-fold X-validation	90%	95.4%	-
	Balance		90%	97.15%	-
[18]	Iris	-	100%	-	98%
	Balance		100%	-	84%
[19]	Car	10x10-fold X-validation	90%	93.3%	-

It can be seen by comparing results in Tables 3 and 4 that proposed method has produced average of 96.44% correct results on the test set of iris flower dataset with the 90% of the training set, which is better than all the methods presented in Table 2. On the balance scale dataset, the proposed method has produced staggering average of 99.76% accurate results on test set with 80% training set, which is again best results among all the contemporary methods. On the car evaluation dataset, the proposed method has produced average of 95.81% correct results against 98.48% average of random forest method [15]. However, random forest method [15] has used 15x10-fold X-validation, whereas proposed method hasn't made any use of x-validation.

6 Conclusion and Future Work

In this paper, a hierarchical nonlinear discriminant classifier is presented. The method proposed automatically produces a tree data-structure according to number of features that can represent nonlinear discriminant classifier based on only four basic mathematical operators $+, -, \times, \div$. The method can retrieve 100% accurate model from the iris flower, balance scale and car evaluation datasets. The example retrieved models are given in appendix. Thus, the model can be useful in knowledge acquisition and decision-making expert systems. Those retrieved models are the array of models placed hierarchically in the model hierarchy list. They can be applied hierarchically to the dataset for the classification purpose. Further, the method is used for classification of the previously unseen data by the model. To achieve this the model was trained on the randomly chosen training set. To classify the test set two models were used i.e. hierarchical application of models present in the model hierarchy list and weighted sum model of all the models present in the model hierarchy list. Weighted sum model produced better results on the iris flower dataset while hierarchical application of models produced better results on the balance scale and car evaluation dataset. Also, the method produced competitive average results when compared with the state of art. Encouraged by these results, now the authors are determined to expand this method to make it applicable to more datasets. Furthermore, detailed analysis is needed to establish why on some datasets weighted sum application of model on the test set performs better than the actual hierarchical application.

Acknowledgments. The authors are grateful for financial support from Innovate UK and Route Monkey Ltd via KTP Partnership number 9839.

Appendix

This appendix contains 100% accurate models that have been retrieved from the iris flower and balanced scale datasets by the proposed computational model when trained on the whole list of samples contained in the datasets. Car evaluation models are not given because of space limitations. Since the method is randomized therefore each run produces different model however, each model when tested back on the datasets produced 100% accurate classification. The models are presented in favour of researchers to help them extract knowledge of the datasets so that they could use this knowledge to build expert systems or develop their own knowledge acquisition and decision-making tools. Table 5 contains these hierarchical models. One model from each dataset is given. For iris flower dataset the model has two hierarchies, while for the balance scale dataset the model has only one hierarchy. Only one model is presented from each dataset because of limited space but there can be many accurate hierarchical models for one dataset, a new model with each run. The beauty of the method is not only in generating many accurate models but also in generating them automatically with only four basic mathematical operators without any mathematical analysis and without any help of analytical tools.

In Table 5, column 1 gives name of the dataset, number of hierarchies in the model is given in column 2, column 3 provides level of model hierarchy, actual trained model in each hierarchy is in column 4, the fitness of model in each hierarchy in terms of number of classified samples is stated in column 5 and finally column 6 shares value of model unfitness or partition wall. Table 1 can be referred to see what feature of dataset corresponding symbols in the model represent.

Table 5. Accurate models of classification datasets

DS	NH	HL	Model description	MF	MUF
(1)	(2)	(3)	(4)	(5)	(6)
Balance scale	1	1	$\frac{0.7813f_4}{0.2571f_2} - \frac{0.9791f_1}{0.3225f_3}$	625	0.00
Iris flower	2	1	$0.9102f_1 - 0.0964f_4 + 0.7035f_2 + 0.0815f_3$	138	0.3439
		2	$\frac{0.5598f_3}{0.0802f_4} + 0.4826f_2 + 0.3099f_1$	12	0.00

Since the models presented in Table 5 are trained on the complete datasets therefore they are based on actual statistical parameters rather than estimated parameters. Therefore, for the development of these models the predictive parameter value in Eq. 6 is taken as $\Delta = 0$. Equation 7 is also replaced to compute actual minimums and maximums for each class member list. There is no need of standard deviation as it was used in Eq. 7 to estimate minimum and maximum value of the model. Procedure-3 should be followed to classify the datasets. The step *d* of procedure-3 i.e., application of relevant model can be broken down as follows in procedure 4.

a. Compute model value for each sample by putting its feature values into the model under consideration.
b. Compute actual mean, minimum and maximum for each class.
c. Compute probability for membership of each sample to each class according to equation 5.
d. Start classification of samples by using relation 9.

Procedure 4: Procedure of application of model

Model Solutions

To give complete sense of the method to readers solutions of above models are presented in Table 6 against one sample from each dataset. In Table 6, column 1 refers to the name of the dataset, level of hierarchy is given in column 2. Column 3 gives class label. For the corresponding class labels Table 2 can be referred. Columns 4–6 provide values of statistical parameters of the corresponding model for each class i.e., minimum, maximum and mean respectively. The feature dimensions of chosen samples are given in column 7. The computed probability of class membership is provided in column 8 and finally column 9 contains the resultant class assigned.

Table 6. Statistical parameters of models for each class of the dataset

Dataset	Hierarchy level	Class	Minimum	Maximum	Mean	Test sample dimensions	Probability of class membership	Class assigned
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
Balance scale	1	c_1	0.0007	0.0181	0.0044	1, 1, 1, 1	0.7927	c_1
		c_2	−14.5709	0.0000	−3.8166		−0.0009	
		c_3	0.1527	14.5898	3.8266		−0.0406	
Iris flower	1	c_1	6.9513	15.1915	10.4918	17, 45, 25, 49	−6.5829	Unclassified
		c_2	44.4055	64.1775	53.3945		0.0046	
		c_3	28.2416	47.2345	39.0089		0.3339	
	2	c_1	0.0000	0.0000	0.0000		$-\infty$	c_2
		c_2	30.5460	32.3623	31.8337		0.0000	
		c_3	32.5227	34.1201	33.0470		−5.7053	

In column 8, with the help of Eq. 5, statistical parameters of models in columns 4–6 are used to estimate class membership probabilities of samples whose dimensions are given in column 7. It can be seen from Table 6 that both the samples, one from each dataset are classified correctly. Please note that sample classification is based on relation 9. Class membership probabilities in column 8 should be used in conjunction with relation 9 to classify the sample. Balance scale model has only one hierarchy therefore it is classified in that hierarchy. The sample of iris flower could not be classified by the model in the first hierarchy. This is because unfitness value in the column-6 of Table 5 prevented it from classifying, as difference in probabilities of class membership was not great enough to surpass unfitness value. It should be noted that if unfitness value would not be there then method would have misclassified the sample as c_3 instead of c_2, as probability of class membership with c_3 has largest value in first hierarchy model. Since the sample remained unclassified in first hierarchy therefore it was tested again in the model in second hierarchy. This model classified it correctly. This is the beauty of hierarchical model that it stops any misclassifications through unfitness value and the sample is given a chance to be classified in the next hierarchy. All the models in the last hierarchy have unfitness value 0.0000. This is done to make sure no sample remains unclassified.

References

1. Bohanec, M., Rajkovic, V.: Knowledge acquisition and explanation for multi-attribute decision making. In: 8th International Workshop on Expert Systems and their Applications, Avignon, France, pp. 59–78 (1988)
2. Bohanec, M., Rajkovic, V.: DEX: an expert system shell for decision support. Sistemica **1** (1), 145–157 (1990)
3. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, Burlington (1993)
4. Ursani, Z., Corne, D.W.: Use of reliability engineering concepts in machine learning for classification. In: 4th International Conference on Soft Computing & Machine Intelligence (IEEE) (ISCMI 2017), Mauritius, November 2017
5. Ursani, Z., Corne, D.W.: A novel nonlinear discriminant classifier trained by an evolutionary algorithm. Accepted in the 10th International Conference on Machine Learning and Computing (ICMLC 2018), University of Macau, China, 26–28 February 2018, ACM Conference Proceedings (2018). ISBN 978-1-4503-6353-2
6. Farid, D.M., Zhang, L., Rahman, C.M., Hossain, M.A., Strachan, R.: Hybrid decision tree and naïve Bayes classifiers for multi-class classification tasks. Expert Syst. Appl. **41**, 1937–1946 (2014)
7. Espejo, P.G., Ventura, 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)
8. Camps-Valls, G., Bruzzone, L.: Kernel-based methods for hyperspectral image classification. IEEE Trans. Geosci. Remote Sens. **43**(6), 1351–1362 (2005)
9. Chao, Y.H., Wang, H.M., Chang, R.C.: A novel characterization of the alternative hypothesis using kernel discriminant analysis for LLR-based speaker verification. Comput. Linguist. Chin. Lang. Process. **12**(3), 255–272 (2007)
10. University of California Irvine Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets.html
11. Anderson, E.: The irises of the Gaspe Peninsula. Bull. Am. Iris Soc. **59**, 2–5 (1935)
12. Fisher, R.A.: The utilization of multiple measurements in taxonomic problems. Ann. Eugen. **7**, 179–188 (1936)
13. Siegler, R.S.: Three aspects of cognitive development. Cogn. Psychol. **8**, 481–520 (1976)
14. Thamano, A., Moolwong, J.: A new computational intelligence technique based on human group formation. Expert Syst. Appl. **37**, 1628–1634 (2010)
15. Mohamed, W.N.H.W., Salleh, M.N.M., Omar, A.H.: A comparative study of reduced error pruning method in decision tree algorithms. In: IEEE International Conference on Control System, Computing and Engineering, Penang, Malaysia, 23–25 November (2012)
16. Kliegr, T., Kuchař, J., Sottara, D., Vojíř, S.: Learning business rules with association rule classifiers. In: Bikakis, A., Fodor, P., Roman, D. (eds.) RuleML 2014. LNCS, vol. 8620, pp. 236–250. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09870-8_18
17. Zhang, L., Ren, Y., Suganthan, P.N.: Instance based random forest with rotated feature space. In: IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL), pp. 31–35 (2013)
18. Ibrahim, S.P.S., Chandran, K.R., Kanthasamy, C.J.K.: Chisc-AC: compact highest subset confidence-based associative classification. Data Sci. J. **13**, 127–137 (2014)
19. Wang, B., Zhang, H.: Probability based metrics for locally weighted naive bayes. In: Kobti, Z., Wu, D. (eds.) AI 2007. LNCS (LNAI), vol. 4509, pp. 180–191. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72665-4_16