Application of Genetic Programming to Induction of Linear Classification Trees

Martijn Bot William B. Langdon

Martijn.Bot@cwi.nl W.B.Langdon@cwi.nl
CWI, Kruislaan 413, 1098 SJ Amsterdam

Abstract

A common problem in datamining is to find accurate classifiers for a dataset. For this purpose, genetic programming (GP) is applied to a benchmark of classification problems. In particular, using GP we are able to induce decision trees with a linear combination of variables in each function node. The effects of techniques as limited error fitness, fitness sharing Pareto scoring and domination Pareto scoring are evaluated. Results indicate that GP can be applied succesfully to classification problems. Comparisons with current state-of-the-art algorithms in machine learning are presented and areas of future research are identified.

1 Introduction

Classification problems form an important area in datamining. For example, a bank may want to classify its clients in good and bad clients or a doctor may want to classify his patients as having diabetes or not. Classifiers may take the form of decision trees [7] (see Figure 1). In the decision trees, in each non-leaf a linear combination of some of the variables is made. This allows classification of continuous and integer valued datasets with an (unknown) inherent linear structure.

When evaluating a decision tree on an individual case from the database, in each non-leaf node, a linear combination is made of some of the available variables. If the sum is smaller than 0, the left classification child of the node is evaluated, otherwise the right child. When finally arrived at a leaf node, a classification is returned for the case. Each function node has the constants and variable numbers in the linear combination as its children. The two rightmost children are either direct classifications or other function nodes. An optimal tree is one which makes as few misclassifications as possible. The problem of finding an optimal decision tree for a given dataset is NP-complete.

Genetic programming (GP) [5] is used as a stochastic search technique for finding accurate decision trees. Genetic programming [5] is a technique inspired by biological evolution and incorporates well-known mechanisms as mutation, crossover and survival of the fittest. A population of individuals exists which compete for survival. Only the most fit ones in each generation will survive and reproduce. Individuals have the form of decision trees (see Figure 1). The fitness of each individual is determined by evaluating it on a dataset and scoring the number

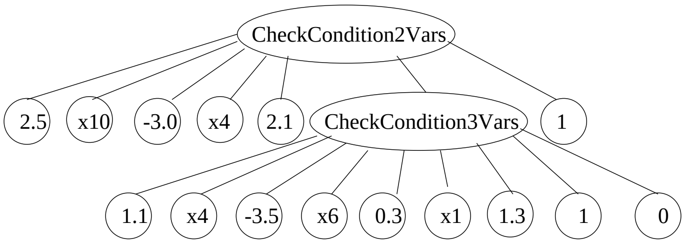

Figure 1: Example decision tree. x_{10} means the tenth variable from the dataset. Each function node's first children are the weights and variables for the linear combination. The last two children are other function nodes or classifications. When evaluating the CheckCondition2Vars node on a certain case, if $2.5x_{10} + -3.0x_4 \leq 2.1$, the CheckCondition3Vars node is evaluated; otherwise the final classification is 1 and the evaluation of the decision tree on this particular case is finished.

of errors. The best individuals (the ones with fewest errors) have the highest chance of reproduction into the next generation. When reproducing, mutation or crossover may occur. When mutating, part of the tree is deleted and a new subtree is created at that point. Crossover takes a subtree from one individual and replaces it with a subtree from another individual. Over the course of evolution, better and better individuals are found. In order to ensure that only valid individuals are constructed in the GP, strong typing was used [8].

The classification accuracy of the GP is compared to that achieved by several other decision tree classification techniques, such as the OC1-algorithm and the C5.0 and the M5' algorithms (Section 5.2).

Three techniques were evaluated. Limited Error Fitness (LEF) is a technique for maintaining more population flux over evolution and reduces runtime (see Section 2.1). Pareto scoring with fitness sharing and Pareto scoring with domination based ranking are techniques for creating and maintaining a diverse population (see Section 2.2). In our system, a population diverse in size and accuracy is created.

In Section 2 the theoretical background behind LEF and Pareto scoring is given. Section 3 explains the experimental setup for the experiments in Section 4. In Section 5 an analysis is made of the performance of the GP-system and a comparison is made to other decision tree algorithms. Section 6 contains our conclusions.

2 Background

2.1 Limited Error Fitness

Limited Error Fitness (LEF) [2] is a modification to the standard supervised learning in GP, where an error limit is introduced. In the process of evaluation of an individual, if the number of errors it has made so far is higher than the error limit, all remaining cases are counted as errors. The fitness score is the total number of errors. This method achieves that poor individuals will not be evaluated on the entire training set, saving CPU time. Good individuals, making fewer errors than the error limit, will be evaluated on the entire training set.

The error limit is modified over time. If no improvement has been made on the best individual in the population for some generations, the error limit is changed. It is raised if the best individual scores more errors than the error limit and lowered if it scores fewer errors than the error limit. Also, the easiest training case(s) (the cases that most individuals classify correctly) is moved to the end of the training set. The most difficult cases are emphasized at the cost of the easier ones, because the easier ones move towards the back.

2.2 Pareto Scoring

Pareto scoring [3] is a way of combining multiple goals in the fitness function. One way of doing this is a linear combination of the different goals. A common problem with this is how to set the weights of the respective goals. Pareto scoring avoids this problem by allowing different combinations of the criteria. If an individual does not perform worse than another individual on every goal, and better in at least one, this individual is said to dominate the other. Individuals can be scored on the number of other individuals they dominate: the more, the better. Those individuals that aren't dominated by any other individual are located on the Pareto optimal front.

Formally, a multi-objective optimization problem is defined a follows:

$$\min_{x \in F} f_1(x), f_2(x), \ldots, f_M(x)$$

where $f_1(x), f_2(x), \ldots, f_M(x)$ are the objective functions to be minimized simultaneously, x is the decision variable and F is the feasible region (the ranges of values allowed for each variable). Solution x_P is on the Pareto optimal front if there exists no $x \in F$ which satisfies

$$\forall m \in 1, \ldots, M : f_m(x) < f_m(x_P)$$

2.2.1 Fitness Sharing

Using Pareto scoring, the population tends to converge to a few (possibly suboptimal) solutions [3]. Fitness sharing (also described in [3]) ensures that more different local minima are found. It is a so-called "crowding technique". To avoid premature convergence, a penalty is given to individuals that have many "neighbours": other individuals that are nearly the same. For a given individual, the degree of sharing is determined by summing the sharing function values contributed by all other individuals in the population (or, for reasons of efficiency, a sample of it). The sharing function is based either on the genotype or on the phenotype of the two individuals. It is triangular, with value 1 for individuals similar in genotype or phenotype and value 0 for individuals less similar.

For each non-dominated individual, the sharing sum is calculated. The one with the lowest sum is the best among them and will be allowed to reproduce. This way, the population itself is used for controlling the evolution process, because the fitness of each individual is not only dependant on the individual itself, but also on other individuals.

2.2.2 Application of Pareto Scoring and Fitness Sharing to minimize size and errors

Pareto scoring will be applied with two dimensions: number of errors (the smaller, the better) and number of nodes in the tree (the smaller, the better). By using fitness sharing, a diverse population with both small trees and good scoring trees can evolve. In the end, the individual which scores fewest errors is returned. Hopefully, this will also be a small tree. Presumably, if such a small and accurate tree exists and is relatively easy to find, the GP will find it.

In [4], genotype-based Pareto scoring and phenotype-based Pareto scoring are distinguished. Genotype-based Pareto scoring compares individuals on the basis of their "genes", i.e. the internal representation. Phenotype-based Pareto scoring compares the solutions produced by the system by various dimensions of the solutions. This system is a combination of the two, as one dimension is genotypic (the number of nodes) and the other is phenotypic (the number of errors).

2.2.3 Domination based Fitness Sharing

An alternative way of fitness sharing [4] is by counting the number of individuals in the population (or the sample) that dominate or are equal in size and errors to each of the non-dominated individuals in the tournament. The individual that is dominated by or equal to the smallest number of individuals wins the tournament and is allowed to reproduce. This is a form of niching, since if there are many individuals which score the same on all dimensions, they will get a bad score. This amounts to a diversive pressure on the population, spreading out the population over the Pareto front.

3 Experimental Setup

We used the GP system by Qureshi (GPSys) [1], which is written in Java. GPSys is a steady state, elitist system with tournament selection. The population size is set at 250, the number of generations is 1000 and the tournament size is 7. The more generations allowed, the more accurate individuals are, so the number of generations was set quite high. The initial population is constructed by the GROW-method from Koza[5]. Ten-fold crossvalidation [7] was used, with 3 runs in each split. After each run, the best individual is reported. Databases were taken from the Machine Learning Repository [2]. The sample size in both Pareto techniques is 50.

[1] `http://www.cs.ucl.ac.uk/staff/A.Qureshi/gpsys.html`
[2] `http://www.ics.uci.edu/~mlearn/MLRepository.html`

Terminals are either constants (doubles), variables (integers) or classifications (integers). In the function nodes, the linear combinations are performed.

After parameter tuning, we found that the exact amount of mutation and crossover was of no high importance, as long as some mutation is present. Thus, the mutation rate was set to 0.5 and the crossover rate also at 0.5. LEF was found to make no difference in accuracy, but since it reduces runtime with approximately 40% (in this implementation), it is used in the following experiments.

4 Experiments

4.1 Pareto Scoring with Fitness Sharing

As explained in Section 2.2, Pareto scoring and fitness sharing are combined in order to find both accurate and small decision trees.

The distance between 2 individuals is based on the shape of the two trees. This is done to create a population diverse in tree shapes and sizes. The Pareto front will range from small, poorly scoring individuals to larger, better scoring individuals. Since a smaller individual with some number of errors would dominate a larger individual with the same number of errors, bloat will be reduced.

The distance between 2 individuals i and j is calculated as follows:

$$d_{i,j} = |\#nodes_i - \#nodes_j| + |depth_i - depth_j|$$

where $\#nodes_i$ is the number of nodes of individual i and $depth_i$ is the depth of individual i. The fitness share is

$$s(d) = \begin{cases} 1 - \frac{d}{10} & d <= 10 \\ 0 & d > 10. \end{cases}$$

The value 10 seemed reasonable since one extra function node adds 5, 7 or 9 extra nodes to the tree (and sometimes increases depth by 1). Note that in the distance measure the number of errors is not included. This is because we don't want to force the system in producing some bad individuals. If the number of nodes is included in the distance measure, the diversive pressure of the fitness sharing allows both well scoring and badly scoring individuals to emerge. For a given tree size, we want the trees to be as accurate as possible.

4.2 Results

In Table 1 an overview is given of the results from the experiments. The average accuracy and tree size of the best individuals from the 30 runs is reported, plus standard deviation.

Experiments were performed to compare the performance of fitness sharing Pareto and domination Pareto. Average accuracy and tree size of the best individuals from each of the 30 runs is given. T-tests were performed to determine if one method produces significantly better or worse validation accuracies.

Domination Pareto creates smaller but less or equally accurate trees than fitness sharing Pareto.

Problem	fitness sharing Pareto		domination Pareto	
	Accuracy	Tree size	Accuracy	Tree size
Glass	63.2 ±10.9	30.4 ±14.4 *	59.1 ±11.5	22.0 ± 7.5
Ionosphere	92.5 ± 4.4	40.9 ±11.4 *	93.4 ± 5.0	26.3 ± 7.7
Pima	75.0 ± 6.7 *	57.5 ±26.3 *	71.6 ± 6.3	39.8 ±16.5
Segmentation	81.7 ± 6.0 *	93.0 ±41.4 *	73.9 ± 6.3	42.3 ±13.3

Table 1: Comparison between fitness sharing Pareto and domination Pareto scoring. Mean validation accuracy of the best individual of the runs plus standard deviation. The stars mark the significantly higher validation accuracies (by a t-test) and significantly larger trees.

5 Analysis

5.1 Pareto Fronts

In Figures 2 and 3 the evolution over time of the Pareto fronts in the population is drawn, with fitness sharing resp. domination Pareto Scoring. With fitness sharing Pareto, improvement can be seen over time over the whole Pareto front. With domination Pareto, the fronts stay more or less the same over time. The only improvements in later generations are made at the very end of the graph (at the highest accuracies).

Because LEF is used in these runs, it can be seen that for example the performance of the smallest individual (which is bad of course) decreases over time. This can be attributed to the error limit, which lowers over time (as better individuals are found), causing the smallest individual to perform worse than before the error limit was lowered.

The zeroth generation contains only few individuals on the Pareto front with the most accurate individual being very large (see the spikes at generation 0). After only a few generations, a much smaller and more accurate individual is found.

5.2 Comparison to Machine Learning Algorithms

In Table 2 the performance of the GP is compared to that of three other decision tree classification algorithms, namely OC1 [9], C5.0 [10] and M5' [1]. Ten-fold crossvalidation is used in the other algorithms.

The GP performs as well as or better than reported decision tree algorithms (OC1, C5.0 and M5') on two datasets (Ionosphere and Pima), but worse on Glass and Segmentation.

6 Conclusions

On some datasets, the GP works as well as or better than reported accuracies for decision tree algorithms, but on others, the GP does worse (see Table 2). More

Problem	GP	OC1	C5.0	M5'
Glass	63.2 ± 10.9	62.3 ± 13.4	67.5 ± 2.6	70.5 ± 2.8 *
Ionosphere	92.5 ± 4.4 *	90.0 ± 5.8	88.9 ± 1.2	89.7 ± 1.2
Pima	75.0 ± 6.7	74.1 ± 6.1	74.5 ± 1.2	76.2 ± 0.8
Segmentation	81.7 ± 6.0	95.4 ± 1.5 *	96.8 ± 0.2 *	97.0 ± 0.2 *

Table 2: Comparison between the GP, random search and the OC1, C5.0 and M5' algorithm. The GP is with fitness sharing Pareto and LEF. Mean training and validation accuracy of the best individual of the runs plus standard deviation. Stars mark significantly better accuracy compared to the GP or better accuracy of the GP.

research is needed to find characteristics of datasets on which the GP does well or poorly. However, genetic programming is a promising approach to classification problems. A disadvantage of this appproach is its long run time: one run on a dataset of only 1000 cases on an UltraSparc 1 takes approximately one hour, while a run of the other techniques typically takes one or two minutes.

As shown in Table 1, fitness sharing Pareto is a promising approach for controlling bloat. It succeeds in creating Pareto fronts ranging from small, inaccurate to large, more accurate individuals (see Figures 2 and 3). Domination Pareto scoring is shown to work worse than fitness sharing Pareto scoring (Table 1).

References

[1] E. Frank, Y. Wang, S. Inglis, G. Holmes, and I.H. Witten. Using model trees for classification. *Machine Learning*, 32:63–76, 1998.

[2] C. Gathercole and P. Ross. Tackling the boolean even N parity problem with genetic programming and limited-error fitness. In *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 119–127, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[3] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Inc., January 1989.

[4] J. Horn and N. Nafpliotis. Multiobjective optimization using the niched pareto genetic algorithm. Technical Report 93005, University of Illinois, July 1993.

[5] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[6] W.B. Langdon, T. Soule, R. Poli, and J.A. Foster. The evolution of size and shape. In L. Spector, W.B. Langdon, U. O'Reilly, and P.J. Angeline, editors, *Advances in Genetic Programming 3*, chapter 8, pages 163–190. MIT Press, Cambridge, MA, USA, May 1999. Forthcoming.

[7] T. Mitchell. *Machine Learning*. WCB/McGraw-Hill, 1997.

[8] D.J. Montana. Strongly typed genetic programming. *Evolutionary Computation*, 3(2):199–230, 1995.

[9] S. Murthy, S. Kasif, S. Salzberg, and R. Beigel. Oc1: Randomized induction of oblique decision trees. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 322–327. AAAI, MIT Press, 1993.

[10] J.R. Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann, 1993.

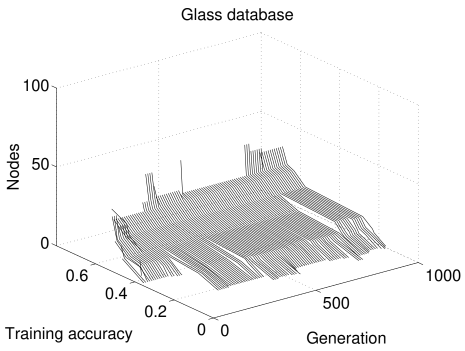

Figure 2: Evolution of Pareto fronts in a run with fitness sharing Pareto on two of the machine learning databases (Glass and Pima).

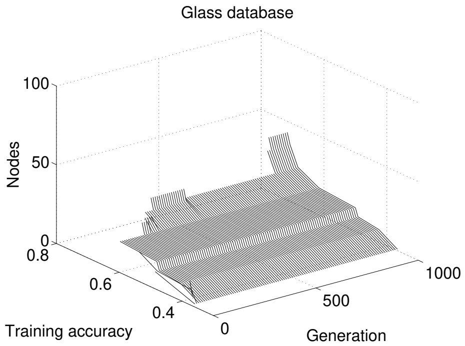

Figure 3: Evolution of Pareto fronts in a run with domination Pareto of two of the machine learning databases (Glass and Pima)