Application of Genetic Programming to Induction of Linear Classification Trees

Martijn C.J. Bot¹ and William B. Langdon²

¹ Vrije Universiteit, De Boelelaan 1081, 1081 HV Amsterdam
mbot@cs.vu.nl
² CWI, Kruislaan 413, 1098 SJ Amsterdam
W.B.Langdon@cwi.nl

Abstract. A common problem in datamining is to find accurate classifiers for a dataset. For this purpose, genetic programming (GP) is applied to a set of benchmark classification problems. Using GP we are able to induce decision trees with a linear combination of variables in each function node. A new representation of decision trees using strong typing in GP is introduced. With this representation it is possible to let the GP classify into any number of classes. Results indicate that GP can be applied successfully 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 credit risks or a doctor may want to classify his patients as having diabetes or not. Classifiers may take the form of decision trees [11] (see Figure 1). In each node, a test is made in which one or more variables is used. Depending on the outcome of the test, the tree is traversed to the left or the right subtree (see Section 2.1). In our decision trees, the tests are linear combinations of some of the variables. This allows classification of continuous and integer valued datasets with an (unknown) inherent linear structure. An optimal tree is one which makes as few misclassifications as possible on the validation set.

Well known decision tree algorithms such as ID3, CART, OC1 and C4.5 are greedy local search algorithms which construct trees top-down [11]. Genetic programming (GP) [5] is used as a global stochastic search technique for finding accurate decision trees. Previous work on evolving decision trees with GP was done in [5] and [12]. The standard representation of GP was used in these experiments. Therefore, the trees look different from most Machine Learning decision trees, where nodes contain linear combinations of variables.

A new representation of decision trees in GP using Strong Typing is introduced. The classification accuracy of the GP is compared to that achieved by several other decision tree classification techniques, such as the OC1-algorithm, C5.0, and the M5' algorithm (Section 5.2).

© Springer-Verlag Berlin Heidelberg 2000
In Section 2 the theoretical background behind the system is given. Section 3 explains the experimental setup for the experiments. The results are given 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.

\[
\begin{align*}
2.5x_{10} - 3.0x_4 & \leq 2.1 \\
1.1x_4 - 3.5x_6 + 0.3x_1 & \leq 1.3 \\
\end{align*}
\]

![Decision Tree Diagram](image)

**Fig. 1.** Example decision tree and its representation in the GP. \(x_{10}\) 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.

## 2 Background

### 2.1 Decision Trees

Decision trees [11] are a well known technique in machine learning for representing the underlying structure of a dataset.

An **axis-parallel** decision tree is one in which each node contains only one variable. All hyperplanes (multi-dimensional planes) are parallel to the axes, hence the name. See Figure 2 for an example. A decision tree is **oblique** when the nodes contain one or more variables. Now the hyperplanes are not necessarily parallel to the axes, but can have any orientation in the attribute space. See