Comparing genetic programming variants for data classification

J. Eggermont [a] A.E. Eiben [ab] J.I. van Hemert [a]

[a] Leiden University, {jeggermo,gusz,jvhemert}@cs.leidenuniv.nl
[b] Free University Amsterdam, gusz@cs.vu.nl

1 Introduction

This article is a combined summary of two papers [1, 2] written by the authors. Binary data classification problems (with exactly two disjoint classes) form an important application area of machine learning techniques, in particular *genetic programming* (GP). In this study we compare a number of different variants of GP applied to such problems whereby we investigate the effect of two significant changes in a fixed GP setup in combination with two different evolutionary models.

2 New features and experiments

A specific representation forms the first line of investigation. In standard GP a function set of real valued operators is used and a special operator in the root of the tree 'Booleanizes' the outcome resulting in a (binary) classification of a given data record. In our atomic representation 'Booleanization' of data is not performed at the root, but at the leafs of a tree and only Boolean functions are used in the tree's body. By such a relatively simple function set the flexibility to create different models, i.e. trees, is lower than in the standard representation, but the models become more transparent, i.e. better readable for humans.

The second modification we consider amounts to using the Stepwise Adaptation of Weights (SAW) technique which has been used in penalty functions for constraint satisfaction problems. Using this technique, each record carries a weight and the total classification error is a weighted sum. During a run the weights are updated periodically: the weights of misclassified records are increased. Hereby the fitness function is changed so that it 'concentrates' more on hard records.

The third aspect we consider concerns steady-state and generational GP models. The standard GP approach is based on the generational population model and although some authors do use a steady-state model, to our knowledge there is no experimental comparison available on the relative advantage of either model.

For experiments we use four different data sets from the Statlog collection[1]. Each algorithm is evaluated using n-fold cross validation and the performance measure for different algorithms is the average classification error over the n-folds.

[1] Available on WWW at `http://www.ncc.up.pt/liacc/ML/statlog`

Statlog uses a cost matrix for two of its data sets. This influences the measured classification error as different penalties are given for each type of misclassification. We have looked at the behavior of the different algorithms both using and leaving out these cost matrices.

3 Conclusions and future research

The comparison of two representations indicates a trade-off situation. Giving up the flexibility of numerical operators for the sake of transparency (using Boolean functions in the bodies of the trees) comes at the cost of performance. It seems that there is no generally advisable option, the choice between the two representations has to be made on a case-by-case basis, depending on the priorities in the given problem context. When we look at the differences between the effects of the SAW-ing mechanism we should distinguish between two cases.

Data sets with and without cost matrix. In case no cost matrix is used GP+SAW clearly outperforms GP alone. This confirms that SAW-ing forms a useful extension of the standard machinery, leading to better results at low costs. On data sets which do use a cost matrix, however, the outcomes are reversed. We hypothesize that this is because a cost matrix biases the measurement of misclassifications which interferes negatively with the re-weighting mechanism of SAW-ing, misleading the GP+SAW algorithm.

The steady-state and the generational models. Using the steady-state model the SAW-ing variant helps the GP to incease performance, but is not able to provide a constant improvement. This is in contrast to the results found in constraint satisfaction and it needs further investigation. Although, one observation shows that SAW-ing causes much fluctuation in the fitness, resulting in a performance decrease when the algorithm is stopped at the wrong time.

Future research will include extending the SAW-ing technique for cost matrices and applying genetic programming to real life data from the financial world.

References

[1] J. Eggermont, A. E. Eiben, and J. I. van Hemert. Adapting the fitness function in GP for data mining. In R. Poli, P. Nordin, W.B. Langdon, and T.C. Fogarty, editors, *Genetic Programming, Proceedings of EuroGP'99*, volume 1598 of *LNCS*, pages 195–204, Goteborg, Sweden, 26–27 May 1999. Springer-Verlag.

[2] J. Eggermont, A. E. Eiben, and J. I. van Hemert. A comparison of genetic programming variants for data classification. In D.J. Hand, J.N. Kok, and M.R. Berthold, editors, *Advances in Intelligent Data Analysis, Third International Symposium, IDA-99*, volume 1642 of *LNCS*, Amsterdam, The Netherlands, 9–11 August 1999. Springer-Verlag.