Bootstrapping to Reduce Bloat and Improve Generalisation in Genetic Programming

Jeannie Fitzgerald
BDS Group
CSIS Department
University of Limerick, Ireland
jeannie.fitzgerald@ul.ie

R. Muhammad Atif Azad
BDS Group
CSIS Department
University of Limerick, Ireland
atif.azad@ul.ie

Conor Ryan
BDS Group
CSIS Department
University of Limerick, Ireland
conor.ryan@ul.ie

ABSTRACT

Typically, the quality of a solution in Genetic Programming (GP) is represented by a score on a given training sample. However, in Machine Learning, we are most interested in estimating the quality of the evolving individuals on unseen data. In this paper, we propose to simulate the effect of unseen data to direct training without actually using additional data, by employing a technique called *bootstrapping* that repeatedly re-samples *with replacement* from the training data and helps estimate sensitivity of the individual in question to small variations across these re-sampled data sets. We minimise this sensitivity, as measured by the *Bootstrap Standard Error*, alongside the training error, in a bid to evolve models that generalise better to the unseen data.

We evaluate the proposed technique on four binary classification problems and compare with a standard GP approach. The results show that for the problems undertaken, the proposed method not only generalises significantly better than standard GP while the training performance improves, but also demonstrates a *strong* side effect of containing the tree sizes.

Categories and Subject Descriptors

1.2.2 [**Artificial Intelligence**]: Automatic Programming - Program Modification

Keywords

Genetic Programming, Binary Classification, Generalisation

In statistics, bootstrapping [2] or *the bootstrap* is a non parametric technique which can provide a confidence interval for some statistic. Bootstrapping also estimates the sensitivity of a statistic of interest to the given data. This measure of sensitivity is called the *Bootstrap Standard Error* (BSE). For example, to estimate BSE of the mean of a data set of size m, we randomly re-sample from the data set with replacement to create n samples, each sample being of size m. Each of the n re-sampled data sets is called a *bootstrap replicate*. Then, for each bootstrap replicate we compute its mean. The BSE, then, is the standard deviation of the means obtained thus from these n bootstrap replicates. Much like standard deviation depicts the variability in any statistic of interest, the lower the BSE, the lower is the variability in the mean of the *original* data set and hence the higher is the confidence in that mean.

For some time, the accepted wisdom in the GP community was that smaller programs may generalise better. This *appeared* consistent with the general principle of Occam's razor, and its formalisation in the minimum description length (MDL) principle proposed by Grünwald [3]. However, in recent years, the link between bloat and over-fitting has become controversial. Recent work by Vanneschi et al.[5] suggests that somewhat contrary to popular belief, these phenomena may be independent to some degree. Azad and Ryan [1] explained that a small GP tree is not necessarily simple: for example, $sin(x)$ is more complex than a larger GP tree encoding $x+x+x+x$.

While this background suggests that the relationship between controlling growth and improving generalisation is not straightforward, both are desirable for GP. This paper reports on a use of bootstrapping with GP which, given the problems undertaken, delivers on both these objectives.

We propose to use bootstrapping to estimate the sensitivity of the evolved models to the training data set. We use this sensitivity as a measure of predictive ability of the evolving solutions on the unseen test data. This treatment is different from previous approaches which formed ensembles from the GP individuals trained over *different* bootstrap replicates derived out of a given training set. In contrast, we use the *entire* training set for every individual in each run. However, after we score an individual on the training set, we also score the individual under consideration on n bootstrap replicates, each derived from the original training set by random resampling *with replacement*. The Bootstrap Standard Error (BSE) then is the standard deviation of these n scores which gives an estimate of the confidence in the original score on the entire training data set. The lower the BSE, the less sensitive is the evolved model to slight variations in the training set; such a model is more likely to generalise to unseen data.

We minimise BSE along with the percentage error rate on the training data for each individual in the population: we compute the fitness of an individual as a product of the corresponding percentage error rate and BSE. The percentage error rate is the percentage classification error of the individual on the overall training set, whereas the BSE is the standard deviation of percentage classification errors on n bootstrap replicates.

Table 1: Fitness Configurations

Acronym	Description	Fitness Function
GP	Standard GP	Error %
BS	BootStrap	Error % * BSE
BSR	BootRandom	Error % * random value: range 0.01-0.061
BSRT	BootTight	Error % * random value: range 0.01-BSE

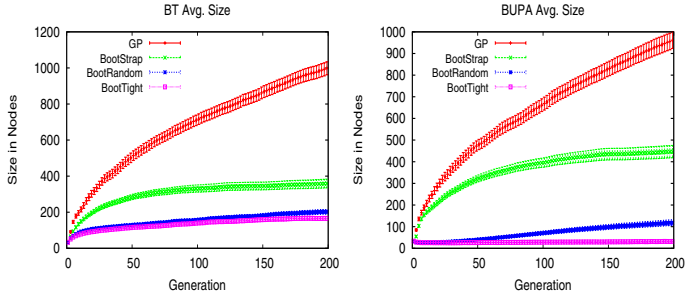
For this investigation, we have experimented with three different bootstrap configurations in addition to a standard GP set-up. Details of the fitness function for each are outlined in table 1. For

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.
ACM 978-1-4503-1964-5/13/07.

Table 2: Best of Run Individuals: All values are averaged over 200 Best-of-run trained Individuals for each task, for each configuration.

Data	Method	Size	Train Err%	Test Err%	Best Test	Gen.
BT	GP	941.82	14.38	23.21	19.50	173.18
	BS	342.05	18.42	22.18	19.21	98.20
	BSRT	136.32	18.63	22.00	19.31	73.47
	BSR	132.46	19.72	22.02	19.31	65.46
BUPA	GP	917.42	9.97	37.63	27.33	176.76
	BS	435.38	19.35	33.95	24.42	119.90
	BSRT	49.56	25.60	33.86	25.00	101.94
	BSR	99.83	24.91	35.49	26.75	129.84
HS	GP	1083.53	11.02	28.48	21.06	156.78
	BS	462.91	17.47	25.16	20.39	115.42
	BSRT	164.45	20.32	25.88	19.74	81.53
	BSR	205.40	20.23	25.35	19.74	97.99
WBC	GP	699.18	0.49	5.28	2.65	103.9
	BS	384.66	0.78	3.72	1.47	102.69
	BSRT	269.95	2.14	4.03	2.06	83.86
	BSR	237.46	2.14	4.26	2.06	120.73

Figure 1: Average Tree Size in Nodes



the standard GP configuration, the fitness measure used to drive the evolutionary process was simply the percentage of misclassified instances, whereas for the proposed Bootstrap method the fitness of each individual was calculated by multiplying the error rate by the BSE of the bootstrap estimates for that individual.

1. RESULTS AND DISCUSSION

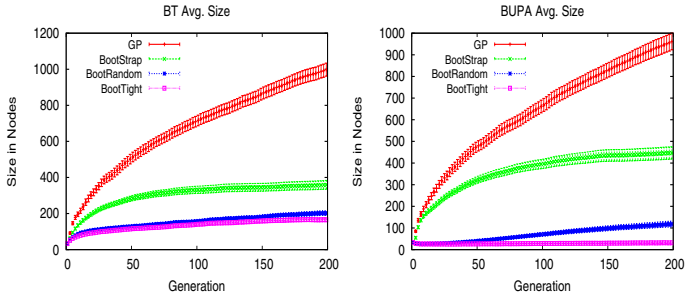
GP out-performed the various BS methods on training data, but on the more important test data the BS method was more competitive. In the case of standard GP, the gap between training and test fitness is larger than with BS, and this gap tends to widen as evolution progresses, suggesting that there is over-fitting occurring with the standard approach which is less evident with BS methods.

We also examined the test performance of the best-of-run *trained individuals* as shown in table 2. Here, we see that the BS method consistently outperformed GP on each of the datasets and that the best trained individual was on average discovered earlier in the evolutionary process using one of the bootstrap configurations.

An interesting (and unexpected) aspect of the results was a dramatic difference in average tree size, as illustrated in Fig. 1. Even though the experiments were constructed *without any explicit growth constraining mechanism*, the average tree size obtained when various bootstrap configurations were used was significantly smaller than that with standard GP. *Without loss of accuracy*, the generation of smaller program trees is a very desirable outcome, offering substantial savings in run times and significant improvements in comprehensibility, particularly when a simple function set is used, as is the case here.

Program growth is seen to taper off at approximately the same time as classification accuracy, on test data when using BS, so there may be a possibility that a consistent reduction in the rate of growth

Figure 2



could be used as an effective early stopping measure, leading to significant time savings, potentially reducing over-fitting, and mitigating the need to guess an appropriate terminating generation.

Given our earlier stated goal of developing classifiers whose training performance could provide a more accurate indication of behaviour on unseen data: training scores achieved using the BS method provided a better indication of the generalisation ability of the programs, as the difference between these outcomes was smaller than when standard GP was used. For the latter experiments, there was over-training on the BUPA and HS datasets, which produced test scores significantly worse than the average training scores, whereas for the BS experiments the average best test scores are at least as good as the average training scores for three of the datasets.

If we accept the definition of bloat in [4] as *program growth without (significant) return in terms of fitness*, we would argue the harmonious tapering off of growth and fitness, strong negative size/fitness correlation and dramatically smaller trees, is evidence that the BS method produces less bloated solutions than GP, without any compromise on test accuracy, and that these solutions may also be less likely to over-fit to the same degree.

2. ACKNOWLEDGEMENTS

This work has been supported by the Science Foundation of Ireland. Grant No. 10/IN.1/I3031.

3. REFERENCES

[1] R. Muhammad Atif Azad and Conor Ryan. Abstract functions and lifetime learning in genetic programming for symbolic regression. In Juergen Branke et al.editors, *GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 893–900, Portland, Oregon, USA, 7-11 July 2010. ACM.

[2] B. Efron and R.J. Tibshirani. *An Introduction to the Bootstrap*. Monographs on Statistics and Applied Probability. Taylor & Francis, 1994.

[3] Peter Grünwald. The minimum description length principle and non-deductive inference. In *Proceedings of the IJCAI Workshop on Abduction and Induction in*, page http://www.mpeg.org/, 1997.

[4] Riccardo Poli, W B Langdon, and Nicholas Freitag McPhee. *A Field Guide to Genetic Programming*. Lulu.com, March 2008.

[5] Leonardo Vanneschi, Mauro Castelli, and Sara Silva. Measuring bloat, overfitting and functional complexity in genetic programming. In Juergen Branke et al. editors, *GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 877–884, Portland, Oregon, USA, 7-11 July 2010. ACM.