Dynamic Training Subset Selection for Supervised Learning in Genetic Programming

Chris Gathercole Peter Ross
Department of Artificial Intelligence
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
U.K.

{chrisg,peter}@aisb.ed.ac.uk

February 25, 1994

Abstract

When using the Genetic Programming (GP) Algorithm on a difficult problem with a large set of training data, a large population size is needed and a very large number of function-tree evaluations must be carried out. This paper describes some efforts made to reduce the number of such evaluations by concentrating on selecting a small subset of the training data set on which to actually carry out the GP algorithm.

Three subset selection methods described in the paper are:
Dynamic Subset Selection (DSS),
using the current GP run to select 'difficult' and/or previously,
Historical Subset Selection (HSS),
selecting 'difficult' cases from previous GP runs,
Random Subset Selection (RSS),
selecting a subset of cases at random for each generation.

Various runs have shown that GP+DSS can produce better results in less than 20% of the time taken by GP. GP+HSS can nearly match the results of GP, and, perhaps surprisingly, GP+RSS can occasionally approach the results of GP.

GP+DSS produced better, more general results than those reported in a paper for a variety of Neural Networks when used on a substantial problem, known as the Thyroid problem.

1 Introduction

At present, the potential of Genetic Programming (GP) and Genetic Algorithms (GAs) has been demonstrated in many different problem areas. Generally, these experiments have involved solving small, relatively neat problems. The future beckons, however, with large and horribly messy problems, to which the GP method will have to be scaled up.

With supervised learning, a training set of cases is involved and the aim is to learn how to classify these known cases and hopefully generalise to be able to correctly classify all possible cases. Large problems will require large training sets. In the standard GP algorithm, the entire population of GP function-trees is evaluated against the entire training data set, and so the number of function-tree evaluations carried out per generation is directly proportional to both the population size and the size of the training set.

In this paper, the simple method of Dynamic Subset Selection (DSS) is described. DSS reduces the number of such evaluations that need to be carried out before a satisfactory answer evolves and, in fact, can produce a more general answer. Two other methods are described for purposes of comparison: the (even) simpler method of Random Subset Selection (RSS), and Historical Subset Selection (HSS) which uses previous GP runs to select a training subset. A classification problem involving the Thyroid data set (described below) was used as a token 'large and messy' problem.

2 The 'Large and Messy' Thyroid Problem

The problem on which most of the experiments were carried out involves the Thyroid data set [Schiffmann *et al* 92a, Schiffmann *et al* 92b, Schiffmann *et al*]. The data is based upon measurements of the thyroid glands of patients at a hospital in Germany. Each measurement vector consists of 15 binary values (0.0 or 1.0) and 6 floating point values (i.e. 21 values in all), and falls into one of three classes. Class 3 signifies a 'normal' thyroid gland and is by far the most common class in both training and test data sets, whilst classes 1 and 2 signify later problems with the gland. To be useful in practise in identifying potential thyroid problems, a classification scheme would have to correctly classify significantly more than 92% of all cases, since over 92% of all patients have a normal (class 3) thyroid gland. There are perhaps between 50 and 100 really difficult cases in the training set and in the test set. So, when attempting to produce a good classification scheme in the range of 97%-100% correct, small improvements are hard to obtain.

There are 3772 cases in the training set and 3428 cases in the test set. Examination of the data in graphical form, e.g. using XGobi [Swayne *et al* 91], reveals that the boundaries between the classes of points are very murky indeed. Points from different classes seem to mingle freely with each other.

In all runs, only the training set is used by the GP to try to evolve its population to classify the thyroid cases into their correct classes. The test set is only used as a check on each generation's best (or fittest) classifier (with respect to the training set), to see how well it generalises to another set of the same kind of data. A run's best classifier is taken to be the one which performs best on the training set. This is not necessarily the one which performs best on the test set.

The setup which generates the fittest classifier with respect to the training set which then performs best on the test set in this way is taken to be the most successful one.

2.1 Modification to Thyroid Problem for this paper

To make things easier for the GP (after a few initial, failed runs), the Thyroid problem was reformulated to classifying cases as class 3 or not class 3. This reformulation allowed the GP function-tree's outputs to be treated as boolean (output $\geq 0 \Rightarrow$ class 3, output $< 0 \Rightarrow$ not class 3) .

It proved relatively straight forward, in a separate run, for DSS to produce a function-tree expression which could distinguish between classes 1 and 2 with 100% accuracy on both the training and test sets. If they were to be used in practise, two GP expression trees would have to be used in two phases: First (and most difficult) distinguish between class 3 cases and the others, then, if it is not a class 3 case, distinguish between class 1 and class 2 cases.

Experiments were carried out with three methods of Subset Selection and compared against the baseline performance of the standard GP which uses the entire training set in each generation.

3 Subset Selection Methods

3.1 Dynamic Subset Selection

This simple idea is based upon a few premises and a small amount of hindsight. Firstly, it is of benefit to focus the GP's attention onto the *difficult* cases, i.e. the ones which are frequently mis-classified. Secondly, it is also of benefit to involve cases which have not been looked at for several generations. This leads to the final point that all of the training cases should be involved, eventually.

The algorithm for DSS involves randomly selecting a target number of cases from the whole training set every generation, with a bias, so that a case is more likely to be selected if it is 'difficult' or has not been selected for several generations.

In each generation, the subset is selected by the following two passes through the full training set.

- In one pass of the entire training set, of size **T**, in a generation, **g**, each training case, **i**, is assigned a weight, **W**, which is the sum of its current 'difficulty', **D**, exponentiated to a certain power, **d**, and the number of generations since it was last selected (or age), **A**, also exponentiated to a certain power, **a**:

$$\forall i : 1 \leq i \leq T, \quad W_i(g) = D_i(g)^d + A_i(g)^a$$

$$\forall i : 1 \leq i \leq T, \quad D_i(g=0) = A_i(g=0) = 0$$

The sum of all the cases' weights is also calculated during this first pass.

- Then, in a second pass of the entire training set, each case in turn has a probability, **P**, of being selected to be in the subset. A case's probability is given by its weight divided by the sum of all the cases' weights and multiplied by the target subset size, **S**:

  $$\forall i : 1 \leq i \leq T, \quad P_i(g) = \frac{W_i(g) * S}{\sum_{j=1}^{T} W_j(g)}$$

  This scaled probability ensures that the average selected subset size is the target size.

  If a case, i, is selected to be in the subset, then its difficulty, D_i, and age, A_i, are set to 0, otherwise its difficulty remains unchanged, and its age, A_i is incremented. While testing each member of the GP population against each case in the current subset of training cases, the difficulty, D_i, (starting from 0) is incremented each time the case is mis-classified by one of the GP trees.

The subset thus consists of all the points selected probabilistically from the training set, and is obviously not always going to be the target size, i.e. it will fluctuate around the target size with each generation. The current generation of the GP's population is then evaluated against this subset of cases instead of the entire training set.

To use this form of DSS, three parameters have to be set:

- the **Target Number** of cases - subset size
- the **Difficulty exponent** - importance given to difficult cases
- the **Age exponent** - importance given to unselected cases

Currently (and, it seems, as always), choosing useful combinations of parameter settings is somewhat of a black art. For the purposes of the Thyroid data set, a target size of 400 (out of 3772) was quickly chosen as an effective value. This corresponds to slightly more than the number of *moderately* 'difficult' cases selected by the HSS method, leaving room for a few easy cases to be included.

With the target size set at 400, it was easier to select sensible values for the two exponents. An average difficulty rating for a case, with a population size of 10000, might be around 2000 or so. The most difficult cases would have a rating of 10000 or so. With a target size of 400, it would take at least 10 generations to cover all the 3772 training cases. With this disparity between a very 'difficult' case and an 'old' case, an arbitrary decision was made to keep the difficulty exponent to 1.0 and to set the age exponent to 3.5 . With these exponents, the most difficult cases and cases around 15 generations old would have roughly equivalent weights.

3.2 Historical Subset Selection

Here, previous standard GP runs are used to establish some measure of how difficult each training case is. Over the course of several runs (say, five or so), the cases mis-classified by the best population member in each generation in each run are recorded. These cases then make up the subset used in further GP+HSS runs, and the subset remains static after its initial selection. Due to the rough-and-ready method by which it is selected, the subset contains a mixture of many 'difficult' points and many which are actually quite 'easy' to classify. Even a best-of-generation population member makes some simple mis-classifications early on in its development.

Some simple checks showed different runs producing very similar subsets selected by this method. The statistics almost always agreed on which cases were most often mis-classified, and only disagreed on some of the easier cases. The subset size used in the runs was 545, and included every single case mis-classified during seven previous runs of a standard GP. A core of around 300 cases were mis-classified more than once or twice, and so were considered to be at least *moderately* 'difficult' cases.

3.3 Random Subset Selection

In this form of selection, at each generation, each case in turn is selected to be in the current subset of training cases with an equal probability. This probability is scaled to ensure that the subset selected, on average, is of the target size.

$$\forall i : 1 \leq i \leq T, \quad P_i(g) = \frac{S}{T}$$

As with the DSS method, the subset size fluctuates around the target size with each generation.

4 GP, Code and Parameters

GP [Koza 92] involves a population of Lisp-like function-tree expressions, each of which is made up from an allowed set of functions and terminals, all of the same data type. As with Genetic Algorithms [Holland 92, Goldberg 89], the GP population is evolved to (hopefully) produce function-trees which can perform well on the problem in question.

The code used for these experiments was adapted from the SGPC code [Tackett & Carmi 93], written in C. The parameter settings chosen here were not optimal, by any means, since the main aim was to demonstrate the effect of *Training Subset Selection* and not *Parameter Tweaking*.

The allowed function and terminal sets used in this paper are:
{ IFLTE, +, -, *, %, TANH, LOG, MINIMUM_OF_3, NEGATE, SQRT },
and { B1 to B15, F1 to F6, 0, -1 }, where 'B' and 'F' refer to the binary and floating point fields of the Thyroid cases.

5 Results

The results given in this paper come from two groups of runs:

- **Population Size = 10000**
  This population size gave the best results achieved by DSS, but, it was not possible to carry out a reasonable length run of the standard GP with this huge population size because it simply took far too long. For the DSS, HSS and RSS runs, the peak values are shown in Table 1 below.

- **Population Size = 5000**
  These results of the GP, DSS, RSS and HSS runs were included to allow a direct comparison of all four methods. Here, the difficulty exponent was increased to 1.5, to allow for the smaller difficulty values.

  For these four runs, the peak values are also shown in Table 1 below, along with the following three graphs:

  - **Figure 1** shows the DSS run easily outperforming RSS.
  - **Figure 2** shows the standard GP run outperforming HSS, though only due to a surge around generation 48. These two methods often produce similar scores, but HSS achieves them with many fewer function-tree evaluations.
  - **Figure 3** shows DSS matching GP results using only 20% of the number of function-tree evaluations.

Table 1: Best Training and Test Results

Algorithm	Pop. Size	Subset Size	Generations	Tree Evals	Training set **% correct**	Test set **% correct**
GP	10000	3772	n/a	n/a	n/a	n/a
GP+DSS	10000	400	69	2.7e+08	99.84	99.27
GP+RSS	10000	400	79	3.2e+08	99.39	98.74
GP+HSS	10000	545	47	2.6e+08	99.20	98.77
GP	5000	3772	60	11.3e+08	99.70	99.00
GP+DSS	5000	400	117	2.3e+08	99.70	99.00
GP+RSS	5000	400	124	2.5e+08	99.10	98.40
GP+HSS	5000	545	57	1.6e+08	99.50	98.70
NN - [Schiffmann *et al* 92a] - Silva and Almeida					99.60	98.45
NN - [Schiffmann *et al* 92a] - Cascade Correlation					100.00	98.48

The best tree produced by the DSS run (with population size = 10000) to distinguish between class 3 and not class 3, was found on Generation 69, giving only 25 errors on the test set. It used only 13 out of the 21 variables available in classifying the Thyroid cases.

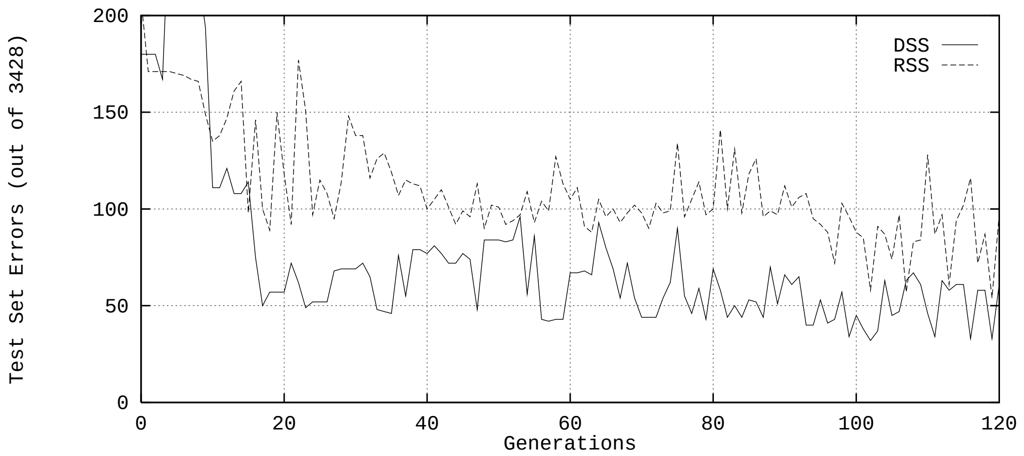

Figure 1: This figure shows the number of errors made on the test set by the best-of-generation trees produced during a run of the DSS and RSS Methods for each generation.

5.1 A quick summary of results from other runs

Different subset sizes were tried, ranging from 50 to the full training set size of 3772. As the subset size approaches 200 or lower, performance drops off markedly and fails to match even that of a standard GP. Much larger subset sizes did not appear to boost performance beyond that of the one of around size 400.

Adding in a parsimony factor (i.e. penalising large and 'bushy' trees) speeds up the running of the GP program, since it then uses much less run-time memory to store the whole population of (smaller) trees, and the trees are quicker to evaluate. The standard GP does not perform as well with this restriction as it does without. However, DSS seemed, if anything, to perform slightly better than before.

6 Discussion

When compared with the Neural Network results in [Schiffmann *et al* 92a], the best of which are shown in Table 1 above, the GP+DSS produced a function-tree which generalised better from the training set. To be fair, in splitting up the problem into two phases (class 3 or not, then class 1 or 2), the GP has been presented with an easier problem than was presented to the Neural Networks. This could be taken in different ways: splitting up the problem is mildly cheating, or demonstrating the flexibility of the GP approach.

The GP + DSS method produces results as good as those of the standard GP and in a much shorter time, on the Thyroid Problem at least. DSS can actually produce better answers, and the population appears to produce a

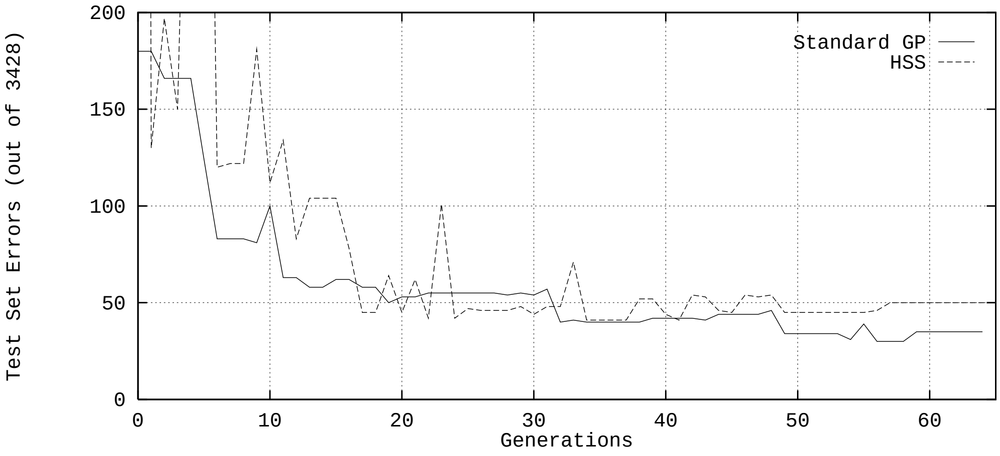

Figure 2: This figure shows the number of errors made on the test set by the best-of-generation trees produced during a run of the Standard GP and HSS Methods for each generation.

larger variety of solutions in later generations than with standard GP or HSS. The random nature of DSS appears to assist the basic GP algorithm.

At this early stage of investigation, there are strong hints that the method is more widely applicable to general problem solving with GPs and GAs involving large training sets (for time saving), and to difficult problems (for better and more general answers). What is more, DSS is easily added to the basic GP algorithm.

HSS out-performed the standard GP in terms of processing time, and nearly matched it in terms of quality of results. HSS was the main contender for *improvement-of-the-week* until DSS was implemented. One big benefit HSS had was the ease with which previous standard GP runs could be cannibalised for information to select the subset of difficult cases.

RSS performs surprisingly well, and can match the performance of standard GP in certain situations, in a much shorter time.

The distribution of errors made by the best function-tree was split more or less evenly between problem cases (classes 1 and 2) and no-problem cases (class 3). This could be altered by biasing the GP algorithm to erring on the side of problem cases, i.e. more False-Positive errors and fewer False-Negative errors, which would be more useful in a medical environment.

Looking at the function-trees produced, it was interesting how the best tree used only used 13 out of the 21 variables available to classify most of the cases correctly. This could perhaps lead to some useful savings in data collection costs, or it could help focus attention on some key measurements. It might be possible to make some further measurements and split each key measurement into several different, finer measurements. One advantage of GP over NNs is that it is very difficult to obtain such insights from a trained NN.

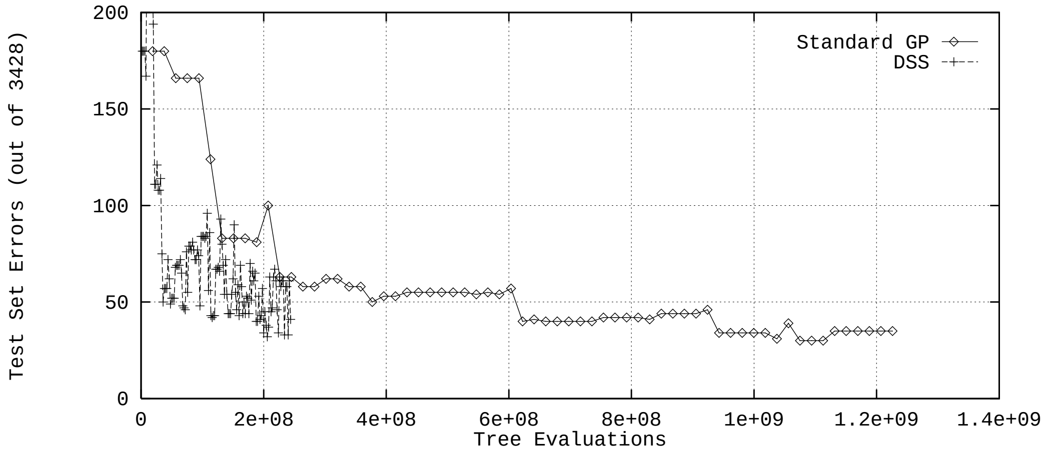

Figure 3: This figure shows number of errors made on the test set by the best-of-generation trees produced during a run of the Standard GP and DSS methods against the number of tree evaluations carried out.

There are myriad lines of investigation to follow up. For instance, how widely applicable is DSS? How does DSS's randomness influence the behaviour of GP? How sensitive is DSS to its parameter settings?

7 Conclusions

Dynamic Subset Selection added to Genetic Programming performs very well on the large, messy Thyroid problem. It produced equivalent answers with around 20% of the number of function-tree evaluations required by a standard GP, when working with a population size of 5000. Working with a larger population size of 10000, DSS produces better answers than any other method shown here, including the NNs.

There are strong hints that DSS is more widely applicable to other supervised training problems with GPs and GAs. Benefits include: much less CPU time needed, better generalisation of the answers, and a very simple algorithm easily added to a standard GA or GP.

Acknowledgments: Thanks are due to Elena Perez-Minana for passing on the Thyroid data and her NN outlook with many helpful discussions, and SERC for my PhD grant Number 93314680

References

[Goldberg 89] David E. Goldberg. *GENETIC ALGORITHMS in Search, Optimisation & Machine Learning.* Addison-Wesley, 1989. Contains clear and detailed descriptions of some Genetic Algorithms.

[Holland 92] John H. Holland. *Adaption in Natural Selection and Artificial Systems.* The MIT Press, 2 edition, 1992. New edition of the original GA work.

[Koza 92] J. Koza. *Genetic Programming: on the programming of computers by natural selection.* MIT Press, Cambridge, MA, 1992. Contains clear description of a basic Genetic Algorithm as well as a detailed description of Genetic Programming.

[Schiffmann *et al*] W. Schiffmann, M. Joost, and R. Werner. Thyroid training and test data sets. Obtained in 1992 via electronic mail from the authors of "Optimization of the Backpropogation Algorithm for Training Multilayer Perceptrons".

[Schiffmann *et al* 92a] W. Schiffmann, M. Joost, and R. Werner. Optimization of the backpropogation algorithm for training multilayer perceptrons. Technical Report 15, University of Koblenz, Institute of Physics, 1992.

[Schiffmann *et al* 92b] W. Schiffmann, M. Joost, and R. Werner. Synthesis and performance analysis of multilayer neural network architectures. Technical Report 16, University of Koblenz, Institute of Physics, 1992.

[Swayne *et al* 91] Deborah Swayne, Dianne Cook, and Andreas Buja. User's manual for XGobi, a dynamic graphics program for data analysis implemented in the X Window System (version 2). Bellcore Technical Memorandum TM ARH-020368, 1991.

[Tackett & Carmi 93] Walter Alden Tackett and Aviram Carmi. S G P C: Simple Genetic Programming in C. Available via FTP at sfi.santafe.edu:pub/Users/tackett, 1993. Original source code for GP program used in this paper.