Evolving Recurrent Models Using Linear GP

Xiao Luo
Dalhousie University
6050 University Avenue
Halifax, NS, Canada
+1 (902) 494-3137
luo@cs.dal.ca

Malcolm I. Heywood
Dalhousie University
6050 University Avenue
Halifax, NS, Canada
+1 (902) 494-2951
mheywood@cs.dal.ca

A. Nur Zincir-Heywood
Dalhousie University
6050 University Avenue
Halifax, NS, Canada
+1 (902) 494-3137
zincir@cs.dal.ca

ABSTRACT
Turing complete Genetic Programming (GP) models introduce the concept of internal state, and therefore have the capacity for identifying interesting temporal properties. Surprisingly, there is little evidence of the application of such models to problems for prediction. An empirical evaluation is made of a simple recurrent linear GP model over standard prediction problems.

Categories and Subject Descriptors
I.2.6 [**Computing Methodologies**]: Learning – *Parameter Learning.*

General Terms
Algorithms, Experimentation, Languages.

Keywords
Recurrent Architectures, Linear Genetic Programming.

1. INTRODUCTION
In this work we are interested in problems for which state needs to be retained across an arbitrary sequence of patterns, hence, pattern sequence is now important. The credit assignment problem has the potential to be much more difficult, with both spatial and temporal dimensions to the learning problem. Leaving a side the definition of appropriate cost functions (e.g. Q-Learning or Temporal Difference methods), there are two basic solutions to building models for solving temporal problems. In the first case the temporal dependence is encoded by the features of each pattern using some *a priori* information, thus reducing the problem to spatial reasoning alone i.e. a supervised learning context. Examples of this might involve encoding the temporal property of the problem using a delay line (shift register) of some predefined depth and resolution. In the second case, a recurrent learning model is employed. This means that the model has capacity to retain state across more than one pattern. Examples of recurrent models include Hidden Markov Models and recurrent neural network models. In both cases support for reasoning about temporal aspects of the problem are provided by feedback paths internal to the model. This makes the learning problem much more difficult as the interconnectivity of these paths requires configuration by the learning algorithm as well as the parameterization of any internal weights. Indeed, various evolutionary approaches have been proposed for building such models [2], [3], [4]. In order to provide an explicitly recurrent Genetic Programming model we note that all that is required is that internal states be retained beyond the current input. Teller previously proposed an indexed memory model within the context of Tree structured GP [5]. To do so, explicit load-store commands were added to the function set. In this work a Linearly structured GP (L-GP) representation is employed. This means that an individual is defined in terms of a sequence of instructions operating on a set of registers (variables). Thus, by not resetting registers until a conclusion state is reached we let GP build the recurrent relationships necessary to reason about time. The specific form of L-GP employed by this work utilizes the page-based L-GP developed in an earlier work [6]. The basic motivation of this work is to compare such a recurrent model against those previously benchmarked on predictive problems. Empirical evidence is provided to demonstrate that recurrent properties can be learnt efficiently.

2. EVALUATION
Two benchmark problems are considered from a recurrent modeling context: Generic solution to the Even Parity problem; and predictor for the Sun Spot series problem. The GP learning parameters are summarized in Table 1. Moreover, these parameters appear to be generic for a cross-section of problems [6].

Table 1. GP Learning Parameters

Dataset	Parity, Sun Spot
Pop. Size	125
Max. Instr.	128
Max Tournaments	50 000
Num. Reg.	4
Function Set	{+, -, *, %}
Terminal Set	{0, .., 255} $\cup$ {input index}
P(Xover)	0.9
P(Mutation)	0.5
P(Swap)	0.9
Runs	50

2.1 Generic Even Parity

The (even) parity problem is a well-known early benchmark in which the basic objective was to derive a specific (even) parity instance using 2 input logical operators that excluded Ex-OR [1]. Here we are interested in deriving 6- and 7-parity from a training set consisting of 2-, 3-, 4- and 5-parity. This results in 252 training sequences and 188 test sequences. A sequence consists of the sequential presentation of each bit associated with the parity case (4-parity having 4 bits etc). On presentation of the last bit in the sequence the value of register R0 is compared to the label for that sequence. From the 50 runs, 24 converged on both training and test set. Computational effort and (un-simplified) instruction count of these cases are summarized in Table 2 using 1st, 2nd (median) and 3rd quartiles. The simplest (and most typical) solution to this problem consisted of two instructions: Ra(t) ← Ra(t – 1) – X(t); R0(t) ← Ra(t) * Ra(t). By all accounts a very concise solution.

Table 2. Quartile Performance on Parity Problem

Quartile	Computational Effort	Instruction Count
1st	369 160	18
2nd (median)	1 723 500	29
3rd	2 489 800	38

2.2 Sun Spot Time Series

The Sun Spot time series has been a benchmark prediction problem in a number of studies [7], [8], [3], [9]. The typical approach has been to use a shift register to provide a sliding window over which a predictive model is built for the next time step i.e. $x(t + 1) = f(x(t), \ldots, x(t - n))$ where 'n' is predefined. In this case the only input is $x(t)$, leaving the selection of relevant previous time steps to the recurrent L-GP model. In line with previous work the dataset is divided into training (221 patterns representing the years 1700-1920), and two test sets (Test set 1 has 35 patterns (1921-1955), Test set 2 has 24 patterns (1956-1979)). Fitness function takes the form of a normalized mean square error,

$$NMSE(P) = \frac{1}{\sigma^2}\frac{1}{P}\sum_{p=1}^{P}\left(desired(p) - GPout(p)\right)^2$$

Where σ^2 = 1 535 and P is the pattern count for the dataset in question [7]. Table 3 provides a comparison against predictors identified in previous works on the same dataset. Of the previous works one other used a (evolved) recurrent model [3] i.e. the other predictors were based on a delay line of predefined depth. The most prominent characteristic here appears to be the consistency of the Recurrent L-GP result over all three partitions. Conversely, the alternative models provide lower errors on training and the first test partition, but degrade significantly on the second test partition representing the period most distant from that used to train the predictors. Table 3 also summarizes complexity of the resulting models. Although no attempt is made to simplify solutions identified by the Recurrent L-GP model it is significantly simpler than the other non-linear models.

Table 3. Comparative Results on Sun Spot Problem

Model	NMSE (train)	NMSE (test1)	NMSE (test2)	Number of Parameters
Recurrent L-GP	0.1077	0.1655	0.1708	35
NN [8]	0.082	0.086	0.35	43
TAR [7]	0.097	0.097	0.28	16
Recurrent NN [3]	0.1006	0.0972	0.4361	22
GP [9]	0.125 ±0.006	0.182 ±0.037	0.37 ±0.06	66.5

3. CONCLUSION

Recurrent Linearly structured GP models have been benchmarked on a series of prediction problems. Central to the ease with which recurrent properties are supported are register (variable) based instruction sets. This is different from providing additional load store instructions for indexing memory. Future work should investigate the utility of active learning algorithms with recurrent L-GP models.

4. REFERENCES

[1] Koza, J.R., *Genetic Programming: ON the Programming of Computers by Means of Natural Selection.* MIT Press, MA, 1992.

[2] Angeline, P.J., Saunders, G.M., Pollack, J.B., An Evolutionary Algorithm that Constructs Recurrent Neural Networks. *IEEE Transactions on Neural Networks*. 5(1), 1994, 54-64.

[3] McDonnell, J.R., Waagen, D, Evolving recurrent Perceptrons for Time-Series Modeling. *IEEE Transactions on Neural Networks*. 5(1), 1994, 24-38.

[4] Xin Yao, Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9), 1997, 1423-1447

[5] Teller, A., The Evolution of Mental Models. In *Advances in Genetic Programming*. Chapter 9. K.E. Kinnear (ed). MIT Press, MA, 1994, 198-219.

[6] Heywood, M.I., Zincir-Heywood, A.N., Dynamic Page-Based Linear Genetic Programming, *IEEE Transactions on Systems, Man and Cybernetics – PartB: Cybernetics*. 32(3), 2002, 380-388.

[7] Tong, H., Lin, K.S., Threshold autoregression, limit cycles and cyclical data. *Journal of the Royal Statistical Society*. B 42, 1980, 245.

[8] Weigend, A.S., Huberman, B.A., Rumelhart, R.E., Predicting the Future: A Connectionist Approach. *International Journal of Neural Systems*. 1(3) 1990, 193-209.

[9] J.C. Principe, L. Wang, Motter M.A., "Local Dynamic Modelling with Self-Organizing Maps and Applications to Nonlinear System Identification and Control," *Proceedings of the IEEE*, 86(11), 1998, 2240-2258.