MACROECONOMICS MODELLING ON UK GDP GROWTH BY NEURAL COMPUTING

Y. Li, K.C. Ng, A. Häußler and V.C.W. Chow

Centre for Systems and Control &
Department of Electronics and Electrical Engineering
University of Glasgow, Glasgow G12 8LT, United Kingdom
E-Mail: Y.Li@elec.gla.ac.uk, Fax: +44 141 330 4907

V.A. Muscatelli

Department of Political Economy
University of Glasgow, Glasgow G12 8RT, United Kingdom

Abstract: This paper presents multilayer neural networks used in UK gross domestic product estimation. These networks are trained by backpropagation and genetic algorithm based methods. Different from backpropagation guided by gradients of the performance, the genetic algorithm directly evaluates the performance of multiple sets of neural networks in parallel and then uses the analysed results to breed new networks that tend to be better suited to the problems in hand. It is shown that this guided evolution leads to globally optimal networks and more accurate results, with less adjustment of the algorithm needed.

Keywords: Economic systems, Modelling, Neural networks, Genetic algorithms, Gross domestic product, Financial systems, Neural computing, Evolutionary computing

1. INTRODUCTION

Traditional models of economic systems are useful in economics studies. These models are not, however, general or adequate enough in explaining many economic anomalies, although they are massively assisted by modern computers. Events such as the "Black Wednesday" of 1992, which forced Britain out of the European exchange rate mechanism, have highlighted concerns over the accuracy in modelling macroeconomic structure and in forecasting the economy. One of the most important factors in macroeconomic modelling and econometrics is the growth, in terms of gross domestic product (GDP). Only when this is accurately modelled and foreseen, can other parameters of the economy, such as unemployment, exports, imports and consumer spending, be reasonably predicted and policies made. These, in turn, affect the GDP, which makes the modelling more complex. Studies show that the vast majority of forecasts on the British economy substantially underpredicted the strength of the economy in the late 1980's and also failed to foresee the depth of the recession that followed (Pain and Britton, 1992).

Clearly, the economic system is not made from mathematical equations and would not, therefore, behave in a similar way to physical systems encountered in engineering. In addition to a stochastic, changing, delayed and nonlinear nature, human involvement by millions of individuals plays a significant role, which results in psychological, fuzzy and learning characteristics inherent in this system. This nature, together with the difficulties encountered in conventional models, has promoted fresh calls for a more "intelligent" modelling methodology that better matches the inherent and underlying characteristics of the economy.

In this paper, neural computing techniques have been applied to modelling the GDP of the United Kingdom. The following section presents artificial neural networks (NNs) used for the modelling, which are trained by "backpropagation (BP)" based techniques. The results are compared with those of a "genetic algorithm (GA)" trained network in Section 3. The final section highlights conclusions and further work.

2. BACKPROPAGATION BASED NEURAL NETWORKS

The information processing system of human beings, who are major players of the economy, is different from conventional computers and mathematics. An example of such a system is the eye-brain system, which consists of many neurons that are processing information and switching at a speed million times slower than a digital computer. Yet, the human system is much more effective and efficient than computers at pattern recognition and at "computationally" complex tasks.

Simulating this human inference and decision making mechanism, the artificial neural network incorporates a learning capability with a parallel information processing structure. Such a network is a physical cellular system that can acquire, store, judge and utilise experimental knowledge and data, which can be inaccurate and incomplete. It has been widely reported its successful implementations and applications to a very broad range of practical problems and, in particular, to modelling of complicated, irregular, irrational, stochastic, nonlinear and time-varying systems (Haykin, 1994; Rogers and Li, 1993). Compared with traditional economic models, an NN features effective means for adaptation and learning in the process of information processing. Further, the smooth nonlinear property of an NN permits the data to fit the model and to generalise better. When realised in software, such an information processing/modelling network is often referred to as "neural computing". This systems behaves rather like a "grey-box" and its use can bypass the step of theory formation that is needed in a traditional economic modelling process. These characteristics make an NN a promising candidate for the GDP and general economic modelling tasks (Hruschka, 1993; Refenes and Azema-Barac 1994; Tafti and Nikbakht 1993).

In this paper, all quarterly data ranging from the first quarter of 1965 to the third quarter of 1994 are obtained from the HM Treasury's Central Statistical Office. These are seasonally-adjusted annualised market prices in £billion. The expenditure based GDP variable is the output of the NN and the consumers' expenditure, general government final consumption, gross domestic fixed capital formation, exports of goods and services and imports of goods and services are used as the input variables for training. In order that an NN can be used correctly to calculate the estimated output, it must be trained from the past data.

Before training takes place, the topology of the NN must be specified. Optimal topologies of an NN, however, vary with types of data and systems to be modelled. They are also dependent upon the associated activation rules and learning mechanism. One of the most popular models is the multilayer network based on the backpropagation training mechanism, as shown in Fig. 1. This model suits time-series tasks more and offers a more accurate function estimation than models of other topologies. It is thus adopted for the time series GDP estimation in this paper.

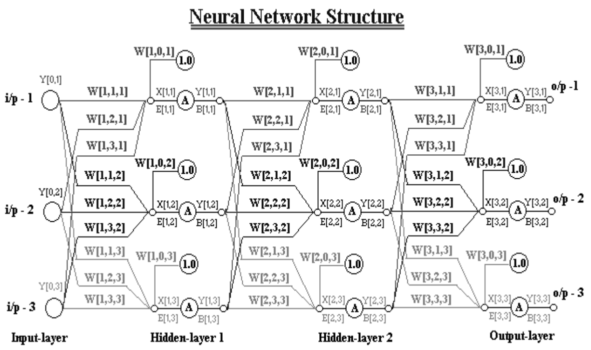

Fig. 1. A multilayer NN with two hidden layers and three input/output neurons.

A multilayer network is essentially a partitioned complete graph which consists of several "layers" of "neurons", where all neurons within a layer can run in parallel whilst all layers run in a pipeline. Every neuron in one layer is not connected to one another but is "completely" connected, with individual "weights", to every neuron in the next layer. A weight can take either positive or negative values. The weighted inputs to a neuron are summed up, usually together with a threshold input, and then the accumulated quantity, x, is filtered through a nonlinear activation function denoted by Ⓐ to form the output of this neuron, as depicted in Fig. 1. The activation function is usually selected either from a sigmoid function, which outputs only positive values, or from a hyperbolic tangent function, which outputs both positive and negative values. The first layer, to which the raw data are fed, is called the "input layer" and the last layer, from which the final estimated outputs are accessed, is called the "output layer". The layers between the input and output layers are termed "hidden layers".

The training of multilayer network can only be carried out when the number of hidden layers and the numbers of their respective neurons are determined. The selection of the architecture is, however, rather a heuristic process, which is mainly dependent upon the size of the problem and the mutual coupling between the variables. In the training process, there are other two factors that also need to be determined. One of them is the "training momentum", which acts as a low-pass filter and allows the network to respond not only to the local gradient, but also to the recent trends in the error surface. The other is the "learning rate", which determines how much the weights should be adjusted in their adaptation process. Details of other network architectures and the backpropagation algorithm can be found in, for example, Haykin (1994). The following summarise the steps needed for training the network.

1. Randomly generate the initial weights and thresholds for the entire network;
2. Feedforward one input data set through the network and calculate the output;
3. Compare the estimated the output with the true data to form the error in the output layer;
4. Compute the error in the last hidden layer;
5. Adjust the weights between the last hidden layer and the output layer based on the amount of error, specified values of learning rate and momentum;
6. Continue the backward error computation and the weight adjustment until the first hidden layer is completed;
7. Go to Step 2 to continue the training until reaching the last training data set;
8. Repeat the training process until a sufficiently small error is reached.

In this section, two different architectures are studied, with comparisons given to the use of one and two hidden layers. In the training phase, data ranging from 1965 to 1992 are used. The estimated GDP by an NN from the five inputs is compared with the real GDP data and then the error is fed back in the backpropagation manner to adjust the weights and minimise the error. A momentum of 0.9 and a learning rate of 0.001 are found to result in relatively fast and effective training in this modelling task. The "incremental method" is used initially in training, which provides a rapid training curve and when the average root mean square (RMS) error falls below 2%, the "batch method" is adaptively switched to, which is slow but fine tunes the network. Once the training is complete, the network is used to estimated the GDP growth for the year 1993 and for the first three quarters of 1994.

2.1 Single hidden layer networks

In this example, an NN with one hidden layer of 10 neurons is tested. Random values of weights are assigned to the two row of interconnections. Here the activation functions used is are hyperbolic tangent functions, whose threshold levels are also trained. The trained and estimated GDP result is depicted in Fig. 2. They are compared with the other curve in the figure, which represents the true GDP data. It can be seen those two curves are close.

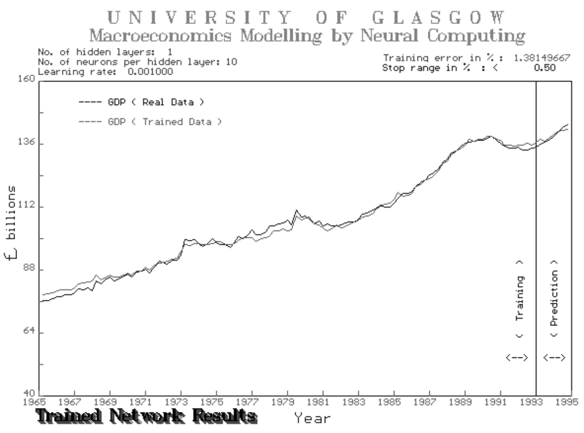

Fig. 2. Comparison between the real GDP and the estimated GDP by an NN with a single hidden layer of 10 neurons.

The training process converges after 1000 iterations (which are termed "epochs") and the average RMS error obtained at the end of 2000 epochs is about 1.38%. This takes about 30 minutes on an Intel 80486DX2 processor running at a clock rate of 66 MHz. Here a Turbo Pascal programme was used in order to have a better flexibility in adaptive learning, instead of using commercially available NN packages.

To compare with this result, sigmoid activation functions are used in another test. Such an NN with the same topology results very similar results to that depicted in Fig. 2. Other architectures of one hidden layer have also been tested but are found not to provide as good estimation as the one with 10 hidden neurons.

2.2 *A double hidden layer network*

In this example, it has been found a network with two hidden layers of 5 neurons resulted in good estimates. The activation functions used here are the hyperbolic tangent functions. The training and estimation results are depicted in Fig. 3. The RMS error at the end of 3000 epochs is about 1.39%. Other double hidden layer based architectures have also been test in this work, but are found not as good as the one reported here.

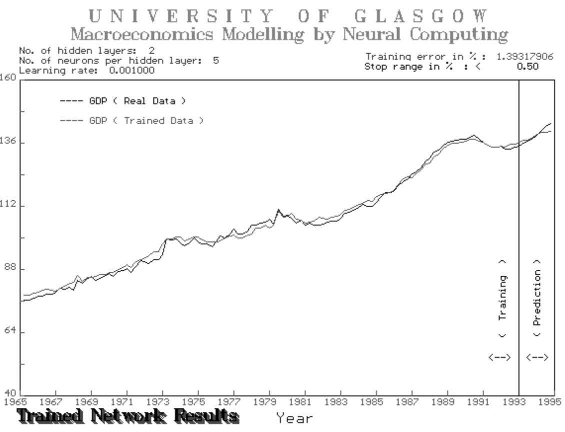

Fig. 3. Comparison between the real GDP and the estimated GDP by an NN with two hidden layer of 5 neurons.

Discussions: In the training tests, it has been found that the BP algorithm lacks the capability of tackling the following problems.

1. Different training processes may result in different minimum average RMS errors for the same large number of epochs. This is because BP is a numerical optimisation method, which is based on "gradient guidance". The search may thus be trapped in "local optima" and be inadequate in finding the "globally optimal" weights and threshold sets;
2. The BP algorithm is difficult to incorporate emphasis on the more recent data and gradually "forget" the more remotely past data. This error-weighting approach is important in time series training, such as the GDP modelling;
3. Finding suitable learning rate and momentum rate is rather a tedious trial-and-error process in BP based training.

In the following section, a genetic algorithm will be used to train the neural network. It will be shown that these problems can easily be tackled by the GA.

3. GENETIC ALGORITHM TRAINED NEURAL NETWORKS

Emulating Darwin's evolutionary principle of "survival-of-the-fittest" in natural selection and genetics, "genetic algorithm" (Goldberg 1989) based search techniques have been found very effective and powerful in searching poorly understood, irregular and complex spaces for optimisation and machine learning. Such an algorithm can simultaneously evaluate performance at multiple points in the solution space and approach the global optima for almost any type of objectives. This technique has been successfully applied to neural network design (Harp and Samad 1992; Yoon, *et al* 1994) and related fuzzy system design (Ng and Li 1994).

In the design of a neural network, a candidate set of all weights and thresholds can be encoded by a binary string. Such a string is termed a "chromosome" in the GA context and a bit of the string is termed a "gene". Initially, many such chromosomes are randomly generated to form a "population". Then the GA uses three basic operators termed reproduction, crossover and mutation to evolve, as depicted in Fig. 4. "Reproduction" is used once the initial population is formed - There are 3 chromosomes in the initial population of the example shown in Fig. 4. As a result of reproduction, a new generation of population is evolved based on their individual "fitness". Here the fitness is a measure of how well a candidate set meets the accuracy specifications. In the designs reported in this paper, the fitness function is given by:

$$f = exp\left(-\frac{1}{\sum_i e_i^2} \right) \tag{1}$$

where *e* is the error between the estimated and true GDP data and *i* the index data. The summation is over the entire training data sets. Since a GA is based on fitness evaluation, no differentiation of this performance function is needed. Thus the formation of this function can be rather relaxed and can incorporate error-weighting factors against time. The relative fitness of an individual in the population is the criterion that determines the probability of its reproduction and survival.

The "crossover" operation in the reproduction process is used to produce some off-spring that inherit a portion of the genetic material of their parents. This is to direct a new search space for further testing within existing domains and is applied to 60% of the entire population in this paper.

In the mean time, "mutation" plays a secondary role in the GA to alter the value of a gene at a random position on the chromosome string, discovering new or restoring lost genetic material. This serves to keep the diversity in the population and searches the neighbouring solution space. It is applied to 5% of the chromosomes in the population. At this point, a new generation is formed and then the process repeats itself until converges to the "fittest" network.

Explanations	Chromosomes	Fitness
Example of coded parameter sets forming an initial population with size 3. The performance of each parameter set is simulated and then assigned a fitness.	C1: 1 1 0 1 0 1 1 0 C2: 1 0 0 1 0 1 1 1 C3: 0 1 0 0 1 0 0 1	f(C1) = 5 % f(C2) = 60 % f(C3) = 35 % (NB. The above fitness values are examples.)
Reproduction: A simple scheme is to allow the chromosomes to reproduce off-spring according to their relative fitness. Thus C1 has low probability of producing children, C2 has a probability of producing two and C3 one.	C2: 1 0 0 1 0 1 1 1 C2: 1 0 0 1 0 ⁞ 1 1 1 C3: 0 1 0 0 1 ⁞ 0 0 1	Evolution in progress (No need to re-calculate fitness here).
Crossover: Some portion of a pair of chromosomes is ex-changed at the dotted position randomly specified.	C2: 1 0 0 1 0 1 1 1 C2':1 0 0 1 0 ⁞ 0 0 1 C3':0 1 0 0 1 ⁞ 1 1 1	No fitness calculations needed here.
Mutation: The binary values of some genes of some chromosomes are inverted. The value which has been changed as an example is highlighted by an under-line.	C2: 1 0 0 1 0 1 1 1 C2": 1 0 $\underline{1}$ 1 0 0 0 1 C3': 0 1 0 0 1 1 1 1	A new generation is now formed and the fitness needs to be evaluated for the next cycle.

Fig. 4. Typical operators of a genetic algorithm.

In this paper, the weights and thresholds of the neural network are coded by decimal strings. This is more direct in mapping the decimal numerals and can smooth, in some degree, the "Hamming Cliff" diversion that is usually encountered in genetic algorithms based on binary coding. For each value of weights and thresholds, three digits are used, which means 1000 possible values of each parameter will be tested in the evolution process. The population size is selected as 100, in view of the size of the network. Note that, however, this and the tasks of selecting the crossover and mutation rates are much easier than determining the learning and momentum rates in BP. The values used here are typical values used elsewhere and are robustly suited to many other applications.

In order to compare with the results obtained from BP, a neural network with one hidden layer of 10 neurons is trained by a genetic algorithm. Fig. 5 depicts the result at the end of 200 generations, which takes about the same time as the BP based training on the same machine running Pascal. The RMS error is 1.47% at this stage, which is slightly larger than the results obtained by BP for a similar running time. Note that, however, the optimisation by BP oscillated around this RMS value and could not converge further. Using the GA, this error can be further minimised by evolving more generations. Fig. 6 shows the convergence curve of the RMS error over 800 generations, at which stage the error reaches 1.02849%.

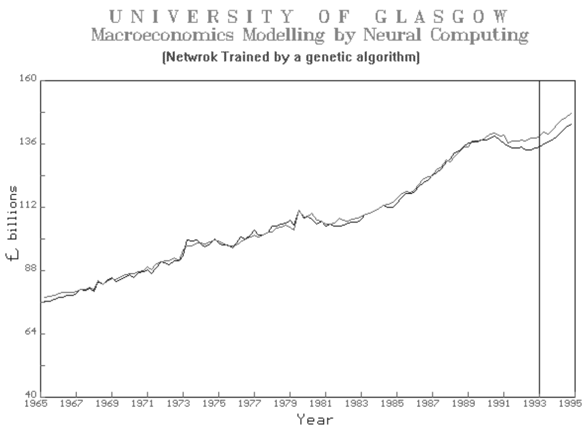

Fig. 5. Comparison between the real GDP and the estimated GDP by an NN with a single hidden layer of 10 neurons trained by a GA.

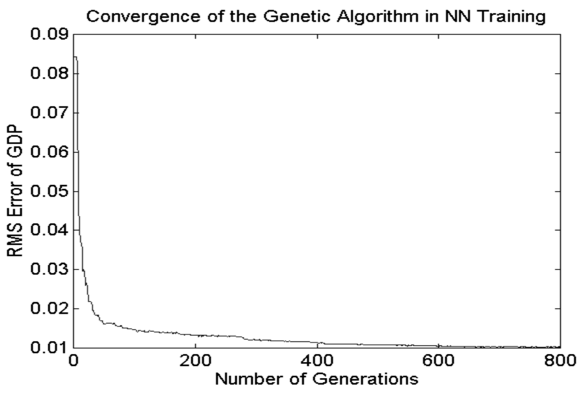

Fig. 6. Convergence curve of the RMS error of the NN trained by a GA.

4. CONCLUSIONS AND FURTHER WORK

This paper has presented backpropagation and genetic algorithm trained neural networks for UK gross domestic product estimation. The BP design mechanism is a conventional optimisation technique based on differentiating the performance index. This imposes restrictions on the properties of the index and could result in locally optimal set of weights and thresholds. Although this may be overcome at some degree by varying the control parameters of the BP algorithm, such as the learning and momentum rates, adaptation of theses parameters is difficult and needs heuristic expertise in setting them. Compared with the BP approach, the GA based training method evaluates the performance of multiple sets of neural networks in parallel. Then the analysed results are used to guide the random search. The survival-of-the-fittest principle evolves globally near-optimal network with better results than the BP based method.

Clearly, other suitable economic variables which directly contribute to the GDP growth, such as the oil price, unemployment rate, inflation rate, interest rates, foreign exchange rates and the stock indices, can be included in the input layer for a better modelling and prediction. The interest rates and foreign exchange rates, for example, may also be used in the output layer for modelling and prediction. This would, however, increase complexity of the problem and the network size. For a reasonable prediction time and for better accuracy, parallel processing systems such as the Parsytec SuperCluster consisting of 64 transputers can be used. This forms part of the future work being considered at Glasgow. Initial studies show that mapping such NNs onto 4 transputers in the SuperCluster will gain a speedup of 3 times (Ng 1992). Currently, incorporating the architecture selections directly in the genetic algorithm is studied at Glasgow. Incorporating new types of neurons, such as those with delay element, is also being considered. In order to reflect the inference nature of human decision making, work on GA based fuzzy systems design (Ng and Li 1994) is extended to fuzzy function estimation to be used for economic system modelling.

Acknowledgement: The authors would like to thank their colleagues in the Evolutionary Computing Group at Glasgow, for useful discussions, and Mr. J. Harding and Mr. A. Hyde of the Databank, Central Statistical Office, HM Treasury, for kindly supplying all the data used in this work. Mr. K.C. Ng is grateful to the University of Glasgow and CVCP for their financial support in the form of a Postgraduate Scholarship and an Overseas Research Scheme Award. Mr. A. Häußler is grateful to University of Darmstadt for offering him an ERASMUS studentship.

REFERENCES

Goldberg, D. (1989). *Genetic algorithms in searching, optimisation and machine learning.* Addison-Wesley, Reading, MA.

Haykin, S. (1994). *Neural Networks*. Paramount Communications Co.

Hruschka, H. (1993). Determining market response functions by neural network modelling: A comparison to econometric techniques. *European Journal of Operational Research,* **66**, 27-35.

Harp, S. A., and T. Samad (1992). Optimizing neural networks with genetic algorithms, *Proc. American Power Conf.*, Chicago IL, **54**, (263) 1138-1143.

Ng, K.C. (1992). *Mapping artificial neural nets onto transputer networks*, Final Year Report, Department of Electronics and Electrical Engineering, University of Glasgow.

Ng, K.C., and Y. Li (1994). Design of sophisticated fuzzy logic controllers using genetic algorithms. *Proc. 3rd IEEE Int. Conf. on Fuzzy Systems*, IEEE World Congress on Computational Intelligence, Orlando, FL, **3**, 1708-1712.

Pain, N., and A. Britton (1992). The recent experience of economic forecasting in Britain: Some lessons from National Institute Forecasts. *National Institute of Economic and Social Research*, **20**, 1-15.

Refenes, A.N., and M. Azema-Barac (1994). Neural network applications in financial asset management. *Neural Computing and Applications*, **2**(1), 13-39.

Rogers, E., and Y. Li (Eds.) (1993). *Parallel Processing in a Control Systems Environment*, Prentice-Hall International (Series on Systems and Control Engineering),.

Tafti, M.H.A., and E. Nikbakht (1993). Neural networks and expert systems: New horizon in business finance applications. *Information Management and Computer Security*, **1**(1), 22-28.

Yoon, B., D.J. Holmes, G. Langholz and A. Kandel (1994). Efficient genetic algorithms for training layered feedforward neural networks. *Information Sciences,* **76**, 67-85.