GENETICALLY OPTIMIZED NEURAL NETWORKS

Francis Wong
NIBS Pte Ltd, 62 Fowlie Rd, Singapore 1542
Geraldine Goh
Accel Infotech (S) Pte Ltd
accel@solomon.technet.sg

1. INTRODUCTION

Recently, neural networks and genetic algorithms have attracted a lot of attention. Neural networks can be regarded, in one respect, as multivariate nonlinear analytical tools, and are known to be very good at recognizing patterns from noisy, complex data, and estimating their nonlinear relationships. However, it is often difficult to design good neural networks. Many basic principles governing information processing in neural networks (such as parallel constraint satisfaction and distributed representation) are hard to understand, and even more difficult to exploit in the design of useful new network architectures. Next, the complex distributed interaction among network units usually makes engineering techniques like divide-and-conquer inapplicable. This complexity also precludes the concoction of direct analytic design methods. Furthermore, it is not possible to guarantee the optimality of a design solution when complex combinations of performance criteria (such as learning speed, compactness, generalization ability, and noise-resistance) are given.

Genetic Algorithm (GA) is very effective at finding optimal or near optimal solutions to a wide variety of problems, because it does not impose many of the limitations required by traditional methods. It is an elegant generate-and-test strategy that can identify and exploit regularities in the environment, and converges on solutions that were globally optimal or nearly so.

Genetic algorithms have been increasingly applied in neural network design in several ways: topology optimization, genetic training algorithms and control parameter optimization. In topology optimization, GA is used to select a topology (number of hidden layers, number of hidden nodes, pattern of connectivity) for the neural network which in turn is trained using some training scheme, most commonly back-propagation. In genetic training algorithms, the learning of a neural network is formulated as a weights optimization problem, usually using the inverse mean squared error as a fitness measure. Many of the control parameters such as learning rate, momentum rate, tolerance level, etc., can also be optimized using GA.

In addition, GA have been used in many other innovative ways, for instance, creating new indicators based on existing ones, selecting good indicators, to evolve optimal trading systemss and to complement other techniques such as fuzzy logic.

2. NEURAL NETWORKS

It has been shown in many work that neural networks are well suited to model dynamical systems. The most popular model used is the backpropagation network and its derivatives. The main mechanism in a backpropagation network is to forward the input data through the hidden layers to the output layers, determine the error between the actual output and the desired output (target), and then back propagate the error from the output to the input through the hidden layers. During the learning phase, the objective is to minimize the global error by modifying the weights associated with the links.

Many studies have shown that neural networks have the capability to learn the underlying mechanics of the time series, or, in the case of trading applications, the market dynamics. In finance and economics we are often confronted with incomplete sets of data. In distinction to technical applications we have normally no chance to redo the measurements. When working with time series data, the standard remedy is to interpolate the lacking data by its neighboring values in the time series. This is a local completion of data. Using neural networks, we can use the global structure of the data to complete the missing values. This method can also be used to build associative question and answer systems : as an input, one can fix some values of a data vector and the system has to complete the vector.

As network applications continue to grow in number, size, and complexity, this human-engineering approach must begin to break down. A more efficient, automated solution is thus needed.

learning problem - Due to an inherent shortsightedness of the learning algorithm of the neural network, the problem of finding a locally good solution instead of a globally optimal one is very real.

Over fitting arises when one attempts to introduce "too many" explanatory variables, too many polynomial terms, or too many lags (in neural terminology, training data extending over too long a time period). In such cases, it is often found that while the additions improve the fit to the past data used for estimation purpose, forecasting performance on new data actually deteriorates, in some cases markedly so.

The standard reaction to this problem is to reduce the amount of data in some way, removing those variables, polynomial terms, or lags, which are thought, or found by some sort of analysis, to be of least value. Many methods for such data filtering have been proposed, ranging from the use in multiple regression of the "adjusted R-squared" rather than the simple R-squared measure of goodness of fit; the Box-Jenkins "parsimonious parameterization" methods; through to the use of the Akaike or Schwarz "information criteria". After the data reduction process, the fitting (training) process is repeated, usually by seeking coefficients (weights) which minimize the sum of squared errors-of-fit over the data retained for estimation.

3. GENETIC ALGORITHMS

There are four stages in the genetic search process: initialization, selection, crossover and mutation. In the initialization stage, a population of genetic structures which are randomly distributed in the solution space is selected as the starting point of the search.

After the initialization stage, each structure is evaluated using a user-defined fitness function and assigned a utility value. On the basis of their relative utility values, structures in the current population are selected for reproduction. A stochastic procedure ensures that the expected number of offspring associated with a given structure s is u(s)/u(P), where u(s) is the observed performance of s and u(P) is the average performance of all structures in the current population. Thus structures with high performance may be chosen for replication while poor-performing structures are removed from the population. In the absence of other mechanisms, such a selective pressure would cause the best performing structures in the initial population to occupy an increasingly larger proportion of the population over time.

The selected structures are recombined using crossover, which two complementary search functions. First, it provides new points for further testing of structures already present in the population; Secondly, it introduces instances of new structures into the population.

Generally, crossover draws only on the information present in the solutions of the current population in generating new solutions for evaluation. If specific information is missing (due to storage limitation or loss incurred during the selection process of a previous generation), then crossover is unable to produce new structures that contain this piece of information. A mutation operator, which arbitrarily alters one or more components of a selected structure, provides the means for introducing new information into the population. However, mutation functions as a background operator with a very low probability of application. The presence of mutation ensures that the probability of reaching any point in the search space is never zero.

4. GENETICALLY OPTIMIZED NEURAL NETWORKS

One method of automating neural network architecture design combines two adaptive processes: genetic search through network architecture space, and backpropagation learning in individual networks to evaluate the selected architectures. Based on Darwinian evolution theory, many possible networks are generated, evaluated, purged and recombined to produce better offsprings which will inherit the knowledge acquired by their parents. Occasionally mutation is introduced to ensure that networks will not be trapped in local minima during the learning process. Networks can be represented by overlapped tree structures. They will respond to the environment and are selected based on the resulting fitness score. The fittest networks reproduce to yield the genotypes in the next generation, and the evolutionary process continues.

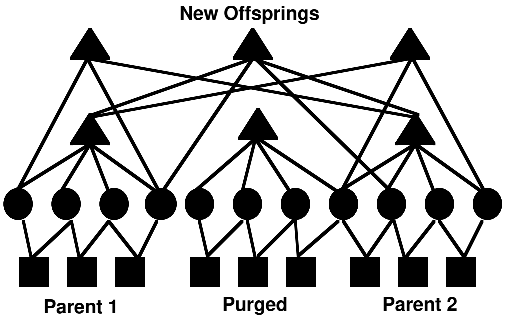

Fig. 1. Mating of two fittest parent networks to give rise to new offsprings which inherit the knowledge acquired by their parents.

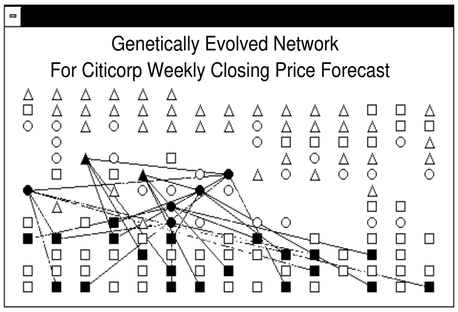

Fig 2. A set of survaving networks after many generations of evolution. The fittest networks are highlighted.

As in real biological systems, learning cycles are nested within cycles of evolution in populations. Each learning cycle involves the entire population of networks with the set of input-output pairs from the input data defining the task, as illustrated in Fig. 3.

The learning algorithm then compares the networks' outputs with the desired outputs, and modifies the connection weights to achieve the desired input-output mapping task more accurately. Each evolution cycle processes one population of network designs according to their associated fitness values (computed during the learning cycles) to yield an offspring population of more highly adapted network designs.

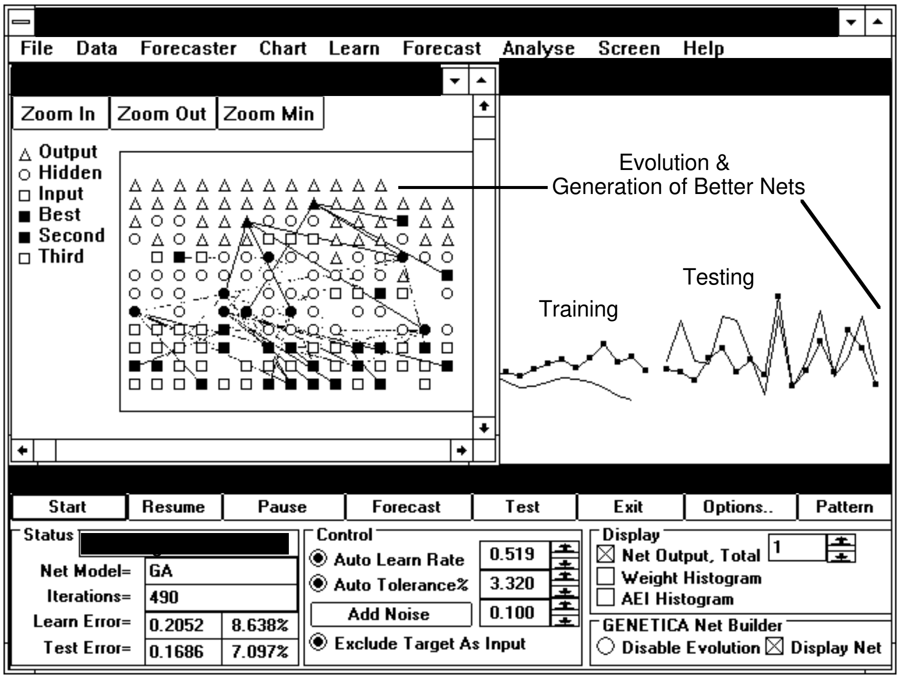

Fig. 3. Training, testing, purging, crossover, mutation, setting of learning rate, tolerance level.

(5) GENETICA - A GA-Based Neural Network Optimization Program

GENETICA is a GA-based neural network optimization module supported by NeuroForecaster version 3.1 and above. In GENETICA, genetic algorithm is applied to search for the best possible neural network topology (pattern of interconnection of neurons, number of input nodes and hidden neurons) and the control parameters (learn rate, momentum rate, tolerance level).

Over fitting or training is another serious problem found in traditional neural networks. Although an over-trained network can fit the past data very well, its forecasting performance on new data generally deteriorates. GENETICA overcomes this problem by selecting the fittest network which give the best performance over the forecasting range, but not over the training range. An image of the best forecaster network is kept and its interconnection weights remain intact while other networks continue to learn the training data. With this selection scheme, the best performance over the forecasting range is always ensured.

GENETICA begins with a population of randomly generated networks, followed by two iterative, adaptive processes as illustrated in Fig. 4. The system genetic search through network architecture space, and back propagation learning in individual networks to evaluate the selected architectures.

Networks are represented by overlapped tree structures. They respond to the environment and are ranked according to their forecasting error. The fittest networks are selected to reproduce the next generation genotypes, and the evolutionary process continues.

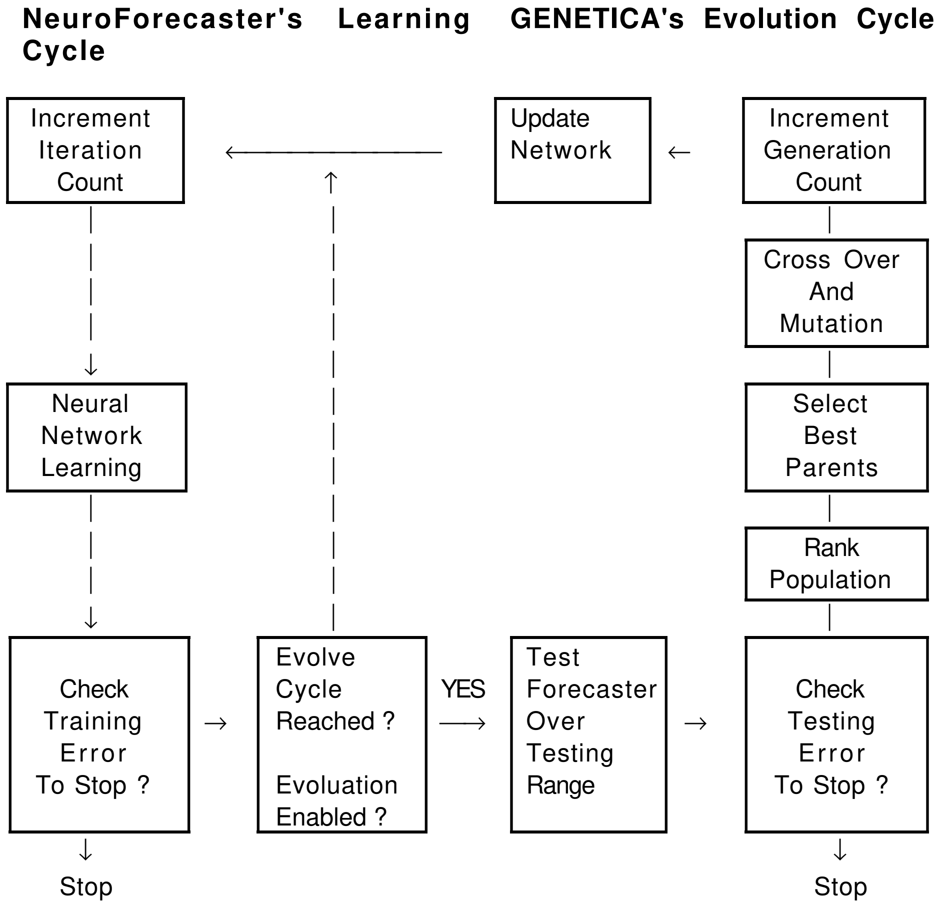

Fig. 4. The integration of NeuroForecaster and GENETICA[1].

The learning algorithm compares the networks' outputs with the desired target, and modifies the connection weights to achieve the desired input-output mapping task more accurately. Each evolution cycle processes one population of network designs according to their associated fitness values (computed during the test) to yield an offspring population of more highly adapted network designs.

---

[1]For more info on NeuroForecaster/Genetica, please email accel@solomon.technet.sg