Web: www.witpress.com Email witpress@witpress.com
Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

Prediction of aerodynamic coefficients for wind tunnel data using a genetic algorithm optimized neural network

T. Rajkumar[1], C. Aragon[2], J. Bardina[2] & R. Britten[3]
[1]*SAIC @ NASA Ames Research Center*
[2]*NASA Ames Research Center*
[3]*QSS @ NASA Ames Research Center*

Abstract

A fast, reliable way of predicting aerodynamic coefficients is produced using a neural network optimized by a genetic algorithm. Basic aerodynamic coefficients (e.g. lift, drag, pitching moment) are modelled as functions of angle of attack and Mach number. The neural network is first trained on a relatively rich set of data from wind tunnel tests or numerical simulations to learn an overall model. Most of the aerodynamic parameters can be well-fitted using polynomial functions. A new set of data, which can be relatively sparse, is then supplied to the network to produce a new model consistent with the previous model and the new data. Because the new model interpolates realistically between the sparse test data points, it is suitable for use in piloted simulations. The genetic algorithm is used to choose a neural network architecture to give best results, avoiding over- and under-fitting of the test data.

1 Introduction

Wind tunnels use scaled models to characterize aerodynamic coefficients. The wind tunnel data, in original form, are unsuitable for use in piloted simulations because data obtained in different wind tunnels with different scale models of the same vehicle are not always consistent. Also, measurements of the same coefficient from two different wind tunnels are usually taken at dissimilar values of the

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

aerodynamic controls (angle of attack, sideslip, etc.), and some means of reconciling the two dissimilar sets of raw data is needed.

Fitting a smooth function through the wind tunnel data results in smooth derivatives of the data. The smooth derivatives are important in performing stability analyses. Traditionally, the approach considered to describe the aerodynamics of the vehicle included developing, wherever possible, a polynomial description of each aerodynamic function [3]. This ensured a smooth continuous function and removed some of the scatter in the wind tunnel data. This curve fitting procedure is unnecessary if the number of coefficients is small. The curve fitting method used to generate the parameters for each polynomial description is an unweighted least squares algorithm. For the most part, the polynomial equations are generated using sparse data from wind tunnel experiments. Because the data is sparse, linear functions are usually employed. When more data are available, flight control system designs need to be revisited to allow for minor nonlinearities in control effects.

An aerodynamic model can be developed from wind tunnel data or by numerical simulation. Wind tunnel testing can be slow and costly due to high personnel overhead and intensive power utilization. Although manual curve fitting can be done, it is highly efficient to use a neural network [4, 19, 21] to describe the complex relationships between variables. Numerical simulation of complex vehicles, on the wide range of conditions required for flight simulation, requires static and dynamic data. Static data at low Mach numbers and angles of attack may be obtained with simpler Euler codes. Static data for stalled vehicles where zones of flow separation are present (usually at higher angles of attack) require Navier-Stokes simulations, which are costly due to the large processing time required to attain convergence. Preliminary dynamic data may be obtained with simpler methods based on correlations and vortex methods [2]; however, accurate prediction of the dynamic coefficients requires complex and costly numerical simulations [20].

This paper is organized as follows: A short introduction to the neural network followed by a section that will introduce the need for optimizing the neural network and basic steps involved in the genetic algorithm. The following section will discuss the aerodynamic data set. Then we discuss the results (finding an optimal solution for the various aerodynamic coefficients). The final section concludes by discussing the benefits of the GA-optimized neural network, initial results and future research directions.

2 Neural network

A neural network is conceptually comprised of a collection of nodes and weighted connections [12, 17, 23]. The initial connection weights are simply random numbers, which change during training. Training consists of presenting actual experimental data to the neural network and using a mathematical algorithm (in this case, the backpropagation algorithm) to adjust the weights. By presenting a sufficient number of input-output pairs, the network can be trained to approximate a function.

Among the many neural network models, the backpropagation algorithm is one

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

of the better known and frequently used. Backpropagation [23] is a generalization of the Widrow-Hoff learning rule [25] to multiple-layer networks and nonlinear differentiable transfer functions. The nodes are arranged in layers, with an input layer and an output layer representing the independent and dependent variable of the function to be learned, and one or more "hidden" layers. Training consists of presenting actual experimental data to the neural network and using a mathematical algorithm – the backpropagation algorithm – to adjust the weights. Each training (input-output) pair of patterns goes through two stages of activation: a forward pass and a backward pass. The forward pass involves presenting a sample input to the network and letting the activations (i.e. node outputs) propagate to the output layer. During the backward pass, the network's actual output (from the forward pass) is compared with the target output and errors are computed for the output units. Adjustments of weights are based on the difference between the correct and computed outputs. The weight adjustments are propagated from the output layer back through previous layers until all have been adjusted (hence "backpropagation").

Feed forward

Apply an input; evaluate the activations a_j and store the error $delta_j$ at each node j:

$$a_j = sum_i(W_{ij}(t)I_i^p)$$
$$A_j^p = g(a_j)$$
$$delta_j = A_j^p - I_j^p$$

After each training pattern I^p is presented, the correction to apply to the weights is proportional to the error. The correction is calculated before the thresholding step, using $err_{ij}(p) = T^p - W_{ij}I^p$.

Backpropagation

Compute the adjustments and update the weights.
$W_{ij}(t+I) = W_{ij}(t) - eta.delta_i.I_j^p$ where $0 \leq eta \leq 1$ is a parameter that controls the learning rate.

- W_{ij} = weight from input i to j in output layer; W_j is the vector of all the weights of the j^{th} neuron in the output layer.

- I^p = input vector (pattern p) = $(I_1^p, I_2^p, ..., I_b^p)$
- T^p = target output vector (pattern p) = $(T_1^p, T_2^p, ..., T_c^p)$
- A^p = actual output vector (pattern p) = $(A_1^p, A_2^p, ..., A_c^p)$
- $g()$ = sigmoid activation function : $g(a) = [1 + exp(-a)]^{-1}$

Each training presentation of the entire set of input-output pairs is called a training "epoch". In general, many epochs of training are required and the error mag-

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

nitude decreases as training proceeds. Once the errors between the intended and actual outputs are within the specified tolerance, training is stopped and the neural network is ready for use: given a new input observation, it will estimate what the corresponding output values should be. After extensive training, the network establishes the input-output relationships through the adjusted weights on the network.

The backpropagation procedure requires that the node transfer functions be differentiable, but importantly, it does not require that they be linear. Typically a hidden layer transfer function is chosen to be nonlinear, allowing extremely complicated relationships to be learned by the network.

3 Need for neural network optimization

The problem of neural network design comes down to searching for an architecture that performs best on some specified task according to explicit performance criteria. This process, in turn, can be viewed as searching the surface defined by levels of trained network performance over the space of possible neural network architectures. Since the number of possible hidden neurons and connections is unbounded, the surface is infinitely large. Since changes in the number of hidden neurons or connections must be discrete, and can have a discontinuous effect on the network's performance, the surface is undifferentiable. The mapping from network design to network performance after learning is indirect, strongly epistatic, and dependent on initial conditions (e.g. random weights), so the surface is complex and noisy [18]. Structurally similar networks can show very different information processing capabilities, so the surface is deceptive; conversely, structurally dissimilar networks can show very similar capabilities, so the surface is multimodal. Hence we seek an automated method for searching the vast, undifferentiable, epistatic, complex, noisy, deceptive, multimodal surface.

The number of nodes on the hidden layer determines a network's ability to learn the intended function from the training data and to generalize it to new data. If a neural network has too many hidden neurons, it will almost exactly learn, or memorize, the training examples, but it will not perform well in recognizing new data after the training process is complete. If a neural network has too few hidden neurons, it will have insufficient memory capacity to learn a complicated function represented by the training examples, i.e. the data will be under-fitted. Training can also be impeded by noise and outliers in the training data. Better convergence can be obtained by simply discarding some training samples, but clearly, this must not be overdone or the correct function will not be learned.

A genetic algorithm is used to optimize the number of training data samples required to train the neural network and the number of hidden neurons in a three layer neural network architecture. The objective of the genetic algorithm is to eliminate training samples that make it difficult for a neural network to converge to the correct output and to avoid discarding data [9, 10, 13, 14]. The fitness function used for the genetic algorithm is chosen to satisfy the conflicting requirements of training-data size reduction. The fitness function for our genetic algorithm per-

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

forms the following calculations for each chromosome in the population:

- Count the number of hidden neurons.
- Count the number of inputs ignored.
- Train the neural network for 500 learning cycles. (Beyond this point, the convergence of the neural network is not very significant.) Sum the training error for the last 40 cycles, to obtain an estimate of overall error in the trained network.
- Calculate the fitness value for a chromosome based on cumulative learning error, the number of inputs that are ignored, and the number of hidden layer neurons.

The fitness function should minimize the training error, the number of hidden neurons and the number of inputs that are ignored (i.e., avoids discarding training cases except when absolutely necessary). In order to optimize the structure of the neural network using a genetic algorithm, a chromosome is encoded using information from input as well as hidden neurons. We chose to use at least 15 neurons, and this value can be encoded in four bits. At least one bit in the chromosome represents information from the input neuron. When a fit chromosome is found, that chromosome is used to specify the number of hidden layer neurons.

4 Genetic algorithm

The basic genetic algorithm comprises four important steps [see [6]] : initialization, evaluation, exploitation (or selection), and exploration.

- The first step is the creation of the initial population of chromosomes either randomly or by perturbing an input chromosome. How the initialization is done is not critical as long as the initial population spans a wide range of variable settings (i.e., has a diverse population). Thus, if explicit knowledge about the system being optimized is available that information can be included in the initial population.
- In the second step, the chromosomes are evaluated and their fitness functions are computed. The goal of the fitness function is to numerically encode the performance of the chromosome. For this problem of optimization, the choice of fitness function is the most critical step.
- The third step is the exploitation or natural selection step. In this step, the chromosomes with the largest fitness scores are placed one or more times into a mating subset in a semi-random fashion. Chromosomes with low fitness scores are removed from the population. There are several methods for performing exploitation. In the binary tournament mating selection method, each chromosome in the population competes for a position in the mating subset. Two chromosomes are drawn at random from the population, the chromosome with the highest fitness score is placed in the mating subset. Both chromosomes are returned to the population and another tournament begins. This procedure continues until the mating subset is full. A characteristic of this scheme is that the worst chromosome in the population will never be selected for inclusion in the mating subset.

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

- The fourth step, exploration, consists of recombination and mutation operators. Two chromosomes (parents) from the mating subset are randomly selected to be mated. The probability that these chromosomes are recombined (mated) is a user-controlled option and is usually set to a high value (e.g., 0.95). If the parents are allowed to mate, a recombination operator is employed to exchange genes between the two parents to produce two children. If they are not allowed to mate, the parents are placed into the next generation unchanged. The two most common recombination operators are the one-point and two-point crossover methods. In the one-point method, a crossover point is selected along the chromosome and the genes up to that point are swapped between the two parents. In the two-point method, two crossover points are selected and the genes between the two points are swapped. The children then replace the parents in the next generation. A third recombination operator, which has recently become quite popular, is the uniform crossover method. In this method, recombination is applied to the individual genes in the chromosome. If crossover is performed, the genes between the parents are swapped and if no crossover is performed the genes are left intact. This crossover method has a higher probability of producing children that are very different than their parents, so the probability of recombination is usually set to a low value (i.e. 0.1). The probability that a mutation will occur is another user-controlled option and is usually set to a low value (e.g., 0.01) so that good chromosomes are not destroyed. A mutation simply changes the value for a particular gene.

After the exploration step, the population is full of newly created chromosomes (children) and steps two through four are repeated. This process continues for a fixed number of generations. For this application, the most widely used binary coded GA is used for encoding genes. In binary coding each chromosome is comprised of zeroes and ones where each bit represents a gene. To formulate the chromosome for optimization, the bit string is concatenated with the bit strings from the other variables to form one long binary string. We adopted a binary coding mechanism for creating the chromosomes. In the next section, we will discuss the data set required for the genetic algorithm optimized neural network.

5 Data set for aerodynamic models

Aerodynamic control systems can be divided into two categories viz., control surfaces and aerodynamic controls. In this paper, aerodynamic controls and models are the focus. The variables involved in aerodynamic controls are angle of attack (α), sideslip angle (β), elevon deflections (δe), aileron deflections (δa), rudder deflection (δR), speed brake deflection (δSB), landing gear effects, and ground effects. The general equations of forces (lb) and moments (ft-lb) for key parameters are listed in the following tables 1 and 2 [3]. The aerodynamic coefficients involved in the above equations are presented.

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

Table 1: Aerodynamic Forces.

Forces (lb)	Model
Lift	$L = CL.q.s$
Drag	$D = CD.q.S$
Side-force	$FY = CY.q.s$

Table 2: Aerodynamic Moments.

Moments (ft-lb)	Model
Pitching	$PM = C_m.q.S.c + (L.cos\alpha + D.sin\alpha).X_{MRC}$ $+(L.sin\alpha - D.cos\alpha).Z_{MRC}$
Rolling	$RM = Cl.q.S.b + FY.Z_{MRC}$
Yawing	$YM = C_n.q.S.b + FY.X_{MRC}$

Longitudinal aerodynamic coefficients

Lift coefficient CL:

$$CL = CL_{BAS}(\alpha, M) + \Delta CL_{,\delta FLAPS}(\delta F) + \Delta CL_{SPEEDBRAKE}(\alpha, \delta SB) + \Delta CL_{LG}(\delta LG) + \Delta CL_{ge}\frac{h}{b} + \Delta CL, q(\alpha, M).q.\frac{c}{2U} + \Delta_{,\alpha\prime}(\alpha, M).\alpha\prime.\frac{c}{2U}$$

Drag coefficient CD:

$$CD = CD_{BAS}(\alpha, M) + \Delta CD_{,\delta FLAPS}(\delta F) + \Delta CD_{SPEEDBRAKE}(\alpha, \delta SB) + \Delta CD_{LG}(\delta LG) + \Delta CD_{ge}\frac{h}{b} + \Delta CD, q(\alpha, M).q.\frac{c}{2U}$$

Pitching moment coefficient Cm:

$$Cm = Cm_{BAS}(\alpha, M) + \Delta Cm_{,\delta FLAPS}(\delta F) + \Delta CLm_{SPEEDBRAKE}(\alpha, \delta SB) + \Delta Cm_{LG}(\delta LG) + \Delta Cm_{ge}\frac{h}{b} + \Delta Cm, q(\alpha, M).q.\frac{c}{2U} + \Delta_{,\alpha\prime}(\alpha, M).\alpha\prime.\frac{c}{2U}$$

Lateral aerodynamic coefficients

Side force coefficient CY:

$$CY = CY_{SB}(\alpha, M).\beta + \Delta CY_{,\delta RUDDER}(\delta R) + \Delta CY_{\delta AILERON}(\delta A)\delta A +$$

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

$$\Delta CY_{LG\Delta\beta}(\delta LG)\beta + \Delta CY_{ge\Delta\beta}\frac{h}{b}\beta + \Delta CY_p(\alpha).p.\frac{b}{2U} + \Delta CY_r(\alpha).r.\frac{b}{2U}$$

Rolling moment coefficient Cl:

$$Cl = Cl_{SB}(\alpha, M).\beta + \Delta Cl_{,\delta RUDDER}(\delta R) + \Delta Cl_{\delta AILERON}(\delta A)\delta A +$$
$$\Delta Cl_{LG\Delta\beta}(\delta LG)\beta + \Delta Cl_{ge\Delta\beta}\frac{h}{b}\beta + \Delta Cl_p(\alpha).p.\frac{b}{2U} + \Delta Cl_r(\alpha).r.\frac{b}{2U}$$

Yawing moment coefficient Cn:

$$Cn = Cn_{SB}(\alpha, M).\beta + \Delta Cn_{,\delta RUDDER}(\delta R) + \Delta Cn_{\delta AILERON}(\delta A)\delta A +$$
$$\Delta Cn_{LG\Delta\beta}(\delta LG)\beta + \Delta Cn_{ge\Delta\beta}\frac{h}{b}\beta + \Delta Cn_p(\alpha).p.\frac{b}{2U} + \Delta Cn_r(\alpha).r.\frac{b}{2U}$$

The above equations depend basically on angle of attack and Mach number with small increments of other factors. The above equations can be expressed as a function of angle of attack and Mach number and they resemble a simple polynomial expression. Depending on the geometry and mass properties of the vehicle, aerodynamic coefficients will vary. The general parameters are tabulated in table 3.
Inputs considered for determining base coefficients are angle of attack and Mach

Table 3: Range of values involved in aerodynamic coefficients.

Parameters	Ranges of values
Angle of attack (degrees)	$-10 < \alpha < 50$
Side angle (degrees)	$-20 < \beta < 20$
Mach number	$M \leq 0.9$
Surface deflection (degrees)	$-15 < \delta elevons(flaps) < 15$ $-20 < \delta rudder < 20$ $-20 < \delta ailerons < 20$ $0 < \delta speedbrake < 80$

number. The outputs of the neural network are the coefficients of the aerodynamic model. As a good training data set for a particular vehicle type, geometry and mass are selected from any wind tunnel test. Sometimes if the data set is not available from wind tunnel experiments, a good training data set can be derived from numerical computations from Euler or Navier-Stokes or Vortex lattice methods. This data set consists of a comprehensive input and output tuple for an entire parameter space.

Once the training data set is defined, sparse data collected from experiments can be interpolated and extended for the entire range of data using a trained neural

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

network (provided the trained data range and sparse data range are similar). This will avoid repeating the entire experiment in the wind tunnel. Once the training data set is selected, one must determine the type of neural network architecture and transfer functions that will be used to interpolate the sparse data. The next section will discuss the selection procedure of the neural network architecture and transfer functions used in this work.

6 Neural network architecture

In this paper, interpolating for coefficient of lift CL is discussed for a sparse data set. (The rest of the various aerodynamic coefficients will be repeated with the same neural network architecture with respect to the corresponding data set.) The problem of defining neural network architectures [8] can be divided into the following categories: (i) type of neural network (whether three layer or four layer, etc.); (ii) number of hidden neurons; (iii) type of transfer functions [5]; (iv) training algorithm; and (v) validation of neural network output, e.g. testing for over- and under-fitting of the results.

If the function consists of a finite number of points, a three layer neural network is capable of learning the function. Additional layers add unnecessary degrees of freedom which may cause the network to over-fit sparse data. Since the availability of data is limited, the type of neural network considered for this problem is a three layer neural network, i.e. input layer, one hidden layer, and output layer. The input layer will have two input neurons (alpha and Mach number) and the output layer will contain a single neuron (coefficient of lift). The data domain has specific definite bounds. The number of hidden neurons is to be chosen based on the efficient fitting of the data.

For determining an appropriate (hopefully optimal or near-optimal) number of hidden units [15], we construct a sequence of networks with increasing number of hidden neurons from 2 to 20. More than 20 hidden neurons cause an over-fitting of the results [16]. Each neuron in the network is fully connected and uses all available input variables. First, a network with a small number of hidden units is trained using random initial weights. Iteratively, a larger network is constructed (up to the 20 hidden neurons) and the network results are compared with the expected results.

Activation functions also play a key role in producing the best network results. The transfer function is a nonlinear function that when applied to the net input of a neuron (i.e. to the weighted sum of its connection inputs), determines the output of the neuron. To get a best fit and characterize physical characteristics of the problem, it is suggested to use different kinds of transfer functions for different layers of the network. The majority of neural networks use a sigmoid function (S-shaped). A sigmoid function is defined as a continuous real-valued function whose domain is the reals, whose derivative is always positive, and whose range is bounded. In this aerodynamic problem, a sigmoid function can produce an efficient fit. However, functions such as "tanh" that produce both positive and negative values tend to yield faster training than functions that produce only positive values such as sig-

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

moid, because of better numerical conditioning. Numerical condition affects the speed and accuracy of most numerical algorithms, and is especially important in the study of neural networks because ill-conditioning is a common cause of slow and inaccurate results from backprop-type algorithms.

The transfer functions for the hidden units are chosen to be nonlinear. (Were they to be linear, the network could realize only linear functions. Because a linear function of linear functions is again a linear function, there would be no value in having a multi-layer network under that condition. It is the capability to to represent nonlinear functions that makes multilayer networks so powerful.) Three types of activation functions are commonly used in backpropagation networks, namely linear, sigmoid and hyperbolic tangent.

The training epoch is restricted to 1000 cycles of: present a data set, measure error, update weights. The learning rate and momentum are selected appropriately to get faster convergence of the network. The input and output values are scaled to the range [0.1, 0.9] to ensure that the output will lie in a region of the nonlinear sigmoid transfer function where the derivative is large enough to facilitate training. The scaling is performed using the following equation:

$$A = r(V - V_{min}) + A_{min}$$

$$r = \frac{A_{max} - A_{min}}{V_{max} - V_{min}}$$

$$V - ObservedVariable$$

$$A - ScaledVariable$$

Once the scaled training data set is prepared, it is ready for neural network training. The Levenberg-Marquardt method [7] for solving the optimization is selected for backpropagation training. It is selected due to its guaranteed convergence to a local minimum, and its numerical robustness. Based on results of the experiments, the neural network is optimized using a binary decoded genetic algorithm.

7 Experiments

The training data set is divided into two sets viz., data set pairs with Mach number less than 0.4, and those greater than 0.4. The data set is presented to the neural network for training. Initially a training set which has 233 pairs is presented to the neural network up to a user-defined error of tolerance. The weights are stored, and a sparse data set of 9 pairs is then provided to the same neural network for further training. The initial training data set represents an exhaustive combination of data points in the particular parameter space, allowing the network to learn the general pattern of a particular aerodynamic coefficient. Based on the general pattern, the second training data set is interpolated.

The initial data set is plotted in figures 1 (a) and (b), and the data in figure 1 (a) can be represented by a linear type of function whereas the data in figure 1

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

(b) can be expressed as a combination of linear and hyperbolic tangent or sigmoid functions. From numerous trials conducted with different combinations of transfer functions, we concluded that the linear transfer function should be adopted for the input-to-hidden neurons and hyperbolic tangent or sigmoid function should be used for the hidden-to-output layer. Figure 1 (c) represents the sparse data set presented to the neural network successively after the initial training data set was presented. The figures 1 (d) and 1 (e) represent the neural network predicted data from the sparse data set. A few points are over-fitted or under-fitted in the results produced by the network. Over- or under-fitting is due to the sparseness of data. Overall the results produced by the network are good.

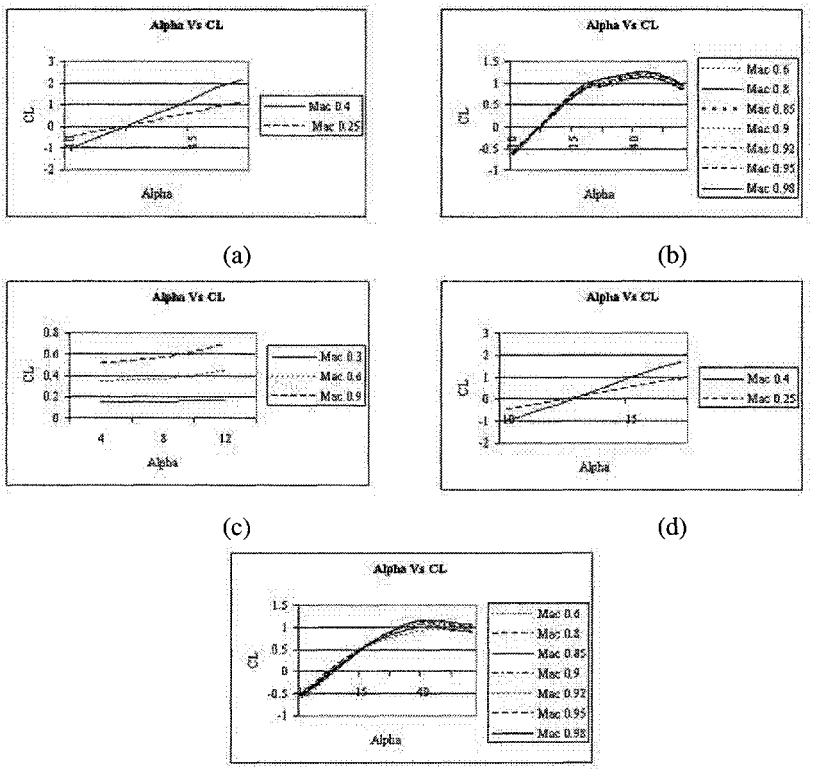
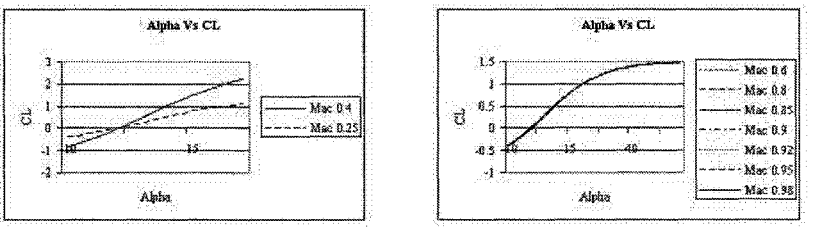


Figure 1: Results from the neural network: (a) Initial training data for neural network ($M \leq 0.4$), (b) Initial training data for neural network ($M > 0.4$), (c) Sparse data presented to the neural network, (d) Neural network interpolated data for sparse data ($M \leq 0.4$) (e) Neural network interpolated data for sparse data ($M > 0.4$).

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9



(a) (b)

Figure 2: GA optimized neural network results: (a) Neural network interpolated data for sparse data ($M \leq 0.4$), (b) Neural network interpolated data for sparse data ($M > 0.4$).

The results which we have shown in the figure 1 are derived using a 2-10-1 neural network architecture. Do these results represent the best prediction that can be obtained by using a neural network? To answer the above question, we use a genetic algorithm to optimize the neural network. The main objectives of optimizing the neural network [1, 24] are (i) Minimum number of training data sets and (ii) Minimum number of hidden neurons. In the first analysis, we presented 233 training pairs for data size reduction. In the initial round of training, 1 data pair was eliminated. In the next round of training, 1 data pair was eliminated. Twenty training cycles were done on the same data. 5 data pairs were eliminated from the original data which were reduced to 228; that is, approximately 0.02% of the total data were eliminated. These data were eliminated because they did not allow a faster convergence for neural network training. Beyond 20 training cycles, no data pairs were eliminated and it took more time for elimination beyond 5 cycles. The next optimization focused on reduction of the number of hidden neurons in the neural network. Based on 228 data pairs, the number of hidden neurons was reduced from 20 (chromosome 5 bit encoding) to 8 hidden neurons. This brings a fast prediction of the coefficient of lift for the aerodynamic model. The final architecture of the neural network is 2-8-1. Figure 2 represents the comparison between the neural network prediction and the genetic algorithm optimized neural network prediction plotted against the measured data. The figure 2 shows substantial improvement of fit when the network is optimized by a genetic algorithm.

8 Conclusion

Neural networks will become an important tool in future NASA Ames efforts to move directly from wind tunnel tests to virtual flight simulations. Many errors can be eliminated, and implementing a neural network can considerably reduce cost. Preliminary results have proven that the neural network is an efficient tool to interpolate across sparse data. The prediction for the lower end and upper end of Mach number by the neural network is considerably deviated. The deviation is

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

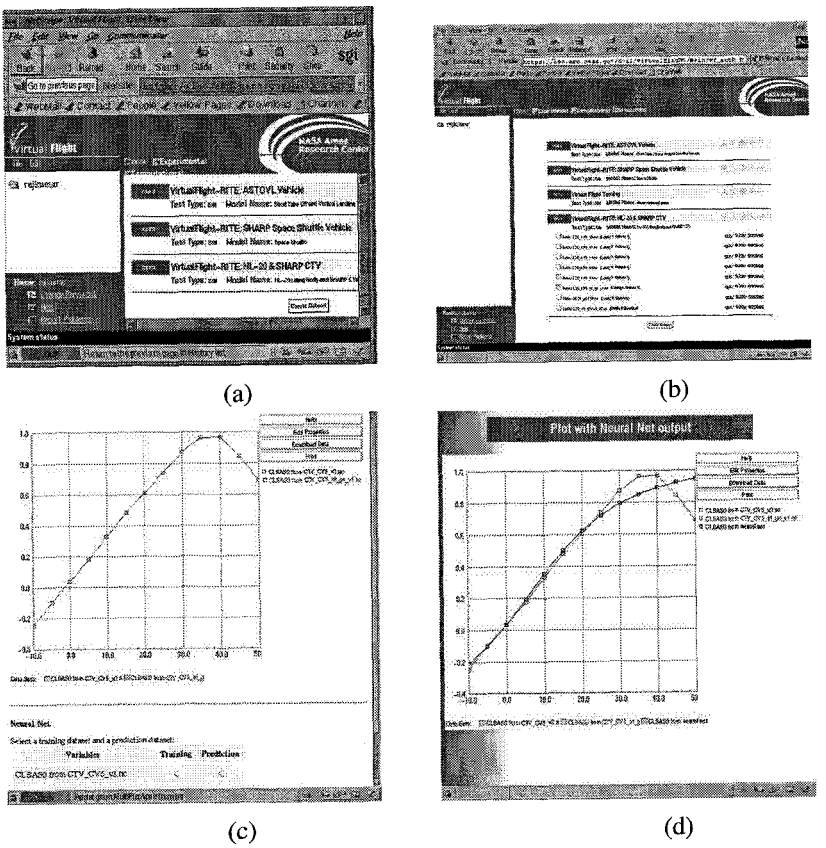


Figure 3: Integration of neural network with DARWIN: (a) Secure DARWIN, (b) Data analysis for virtual flight in DARWIN , (c) Training data definition in DARWIN, (d) Neural network prediction.

caused by non-availability of data in the sparse set. Initially the neural network has been trained by original data which enables the network to understand an overall pattern. Successive training by the sparse data alters the weights of the neural network which causes this deviation. This deviation is well within 10 %, which is acceptable in aerodynamic modelling.

When not much is known about the response surface and computing the gradient is either computationally intensive or numerically unstable, a genetic algorithm is efficient. In our problem, we would like to get an optimized neural network architecture and minimum data set. This has been accomplished within 500 training cycles of a neural network. Repeated execution of a neural network takes 8 hours to create the optimized data set and architecture. The neural network constructed is a feed forward neural network with a back propagation learning mechanism. The

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

main goal has been to free the network design process from constraints of human biases, and to discover better forms of neural network architectures. The automation of the network architecture search by genetic algorithms seems to have been the best way to achieve this goal. A hybrid system using evolutionary theory and a neural network is planned to build an efficient model to predict aerodynamic variables. The neural network will be an integral tool of the data mining suite in an existing collaborative system (DARWIN) at NASA. DARWIN is a collaborative tool for scientific researchers at NASA Ames. The neural network is already integrated and screen snap shots are shown in figure 3 (a-d). Further research is planned to integrate the genetic algorithm into DARWIN and allow the user the freedom to alter various variables in the neural network and genetic algorithm to customize their models.

References

[1] Bigus, P.J. & Bigus, J., *Constructing intelligent agents using java*, Wiley computer publishing, 2001.
[2] Brickelbaw, L.AG., McNeill, W.E. & Wardewell, D.A., Aerodynamics model for a generic STOV LIFT-FAN AIRCRAFT. *NASA: TM 110347*, USA, 1995.
[3] Bruce, J. & Christopher, C., Preliminary subsonic aerodynamic model for simulation studies of the HL-20 lifting body. *NASA:TM 4302*, USA, 1992.
[4] Ching, F.L., Zhao, J.L. & DeLoach, R., Application of neural networks to wind tunnel data response surface methods. *Proc. of the 21st AIAA aerodynamic measurement technology and ground testing conference* , Denver, USA, 2000.
[5] Elliott, D.L., A better activation function for artificial neural networks. *ISR Technical Report 93-8*, 1993.
[6] Goldberg, D.E., *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley, 1988.
[7] Hagan, M.T. & Menhaj, M.B., Training feed forward networks with the Marquardt algorithm. *IEEE transactions on neural networks*, **Vol 5, No 6**, pp 989-993, 1994.
[8] Hagan, M.T., Demuth, H.B. & Beale, M.H., (eds), *Neural network design*, PWS Publishing: Boston, USA, 1996.
[9] Hancock, P.J.B., Pruning neural nets by genetic algorithm. *Proc. of the international conference on artificial neural networks*, Brighton, 1992.
[10] Hochman, R., Khoshgoftaar, T.M., Allen, E.B. & Hudepohl, J.P., Using the genetic algorithm to build optimal neural networks for fault-prone module detection. *Proc. of the 7th international symposium on software reliability engineering (ISSRE '96)* IEEE press, 1996.
[11] Holland, J.H., *Adaptation in natural and artificial systems*, Ann Abor, MI:University of Michigan press, USA, 1975.
[12] Jondarr, C., Backpropagation family. *Technical Report C/TR96-05 Macquire university*, 1996.
[13] Jurgen, B., Evolutionary algorithms for neural network design and training.

Paper from: *Data Mining III*, A Zanasi, CA Brebbia, NFF Ebecken & P Melli (Editors).
ISBN 1-85312-925-9

Proc. of the 1st Nordic workshop on genetic algorithms and its applications, Vasa, Finaland, 1995.

[14] Kroning, P.G., Training of neural networks by means of genetic algorithm working on very long chromosomes. *Technical Report Computer Science department*, Aarhus, Denmark, 1994.

[15] Lawrence, S., Lee, G., & Ah Chung, T., What size neural network gives optimal generalisation? Convergence properties of Backpropagation. *Technical Report UMIACS-TR-96-22 and CS-TR-3617*, 1996.

[16] Lawrence, S., Lee, G. & Ah Chung, T., Lessons in neural network training : Overfitting may be harder than expected. *Proc. of the 14th national conference on artificial intelligence, AAAI-97*, USA, 1997.

[17] Master, T., (eds), *Practical neural network recipes in C++*, Academic Press Inc.: USA, 1993.

[18] Miller, G., Todd, P. & Hegde, S., Designing neural networks using genetic algorithms. *Proc. of the 3rd international conference on genetic algorithms*, George Mason university,USA, 1989.

[19] Norgaard, M., Jorgensen, C. & Ross, J., Neural network prediction of new aircraft design coefficients. *NASA: TM 112197*, USA, 1997.

[20] Park, M.A. & Green, L.L., Steady-state computation of Constant rotational rate dynamic stability derivatives. *AIAA* , USA, 2000.

[21] Rai, M.M. & Madavan, N.K., Aerodynamic design using neural networks. *AIAA*, **Vol 38, No 61**, pp 173-182, 2000.

[22] Rajkumar, T. & Bardina, J., Prediction of aerodynamic coefficients using neural networks for sparse data. *FLAIRS 2002 conference*, Pensacola, USA, 2002.

[23] Rumelhart, D.E., Hinton, G.E. & Williams, R.J., Learning internal representations by error propagation. *Parallel data processing*, **Vol 1**, Cambridge, MA: The MIT press pp 318-362, 1986.

[24] Watson, M., *Intelligent Java applications*, Morgan Kaufmann publishers: USA, 1997.

[25] Widrow, B., Winter, & Baxter, Learning phenomena in layered neural networks. *Proc. of the 1st international conference on neural nets*, pp 411-429, Sandiego, USA, 1987.