International Journal of Advanced Science and Technology
Vol. 34, September, 2011

Gradient Descent Feed Forward Neural Networks for Forecasting the Trajectories

Anurag Sharma

Department of Computer Science, Singhania University, Chittorgrah, Rajsthan, India
Anurag.shamli@gmail.com

Ashish Chaturvedi

Department of Applied Sciences, Gyan Bharti Institute of Technology, Meerut, UP, India
E-mail: dr_ashysh@yahoo.com

Abstract

The paper demonstrates the forecasting of an aircraft trajectory in the vertical plane using gradient descent method for training a feed forward neural network system. For prediction of trajectory a neural networks system has been trained using a set of some arbitrary trajectories and then used to forecast for the new ones. Sliding Window method is being used for predictions, which is able to consider real points during flight to improve the precision in prediction. The results show that neural network can successfully be applied for such predictions.

Key words: *Gradient descent networks, forecasting of trajectories.*

1. Introduction

Volatile expansion of commercial airways causes the increasing delays in flights and thus demands for the management of air traffic trajectories. Now a day Air Traffic Control (ATC) includes Trajectory prediction as one of the most significant problems to be considered. All the control systems depend on the quality of the prediction. The predictions in two dimensions (x, y) may accurately be measured. However, the predictions in three dimensions (x, y, t) are still not accurate. Heading can be forecasted precisely but speed prediction undergoes a lack of precision. In Past few years, many different simulations have been done using flight equations; creating models of aircraft [1- 3] in order to simulate flights. These models are either based on general aerodynamic equations or tables giving speed corresponding to the current altitude; and using non-parametric methods [4].

The main problem with such methods is that they need some parameters that are not easy to get just like, vertical speed depends on various parameters such as the aircraft take off weight; thrust, drag and lift are functions of the aircraft type, of flight parameters given to the Flight Management Systems etc. such information's are usually unavailable to the ground control system. Our idea is to predict aircraft trajectories from a reduced set of known points using neural networks. If we have a set of initial radar plots of an aircraft, then our aim should to forecast its future trajectory. The method must be independent of parameters like weight, wind, operator's flight procedure or flight plans.

2. Gradient Descent Learning Algorithm of Feed Forward Neural Networks

The Gradient descent back-propagation algorithm is a gradient descent method minimizing the mean square error between the actual and target output of a multilayer perceptron. Assuming sigmoidal nonlinear function

$$f(net_i) = \frac{1}{1-e^{-net}} \qquad (5)$$

The back-propagation algorithm consists of the following steps:

I. Initialize Weights and Offsets

Initialize all weights and node offsets to small random values.

II. Present Input and Desired Output Vector

Present continuous input vector **x** and specify the desired output **d**. Output vector elements are set to zero values except for that corresponding to the class of the current input.

III. Calculate Actual Outputs

Calculate the actual output vector **y** using the sigmoidal nonlinearity.

IV. Adapt weights

Adjust weights by

$$w_{ij}(t+1) = w_{ij}(t) + \iota \delta_j x_i^{'} \qquad (6)$$

where is the output of the node i and is the sensitivity of the node j. If node j is an output node, then

$$\delta_j = f'(net_j)(d_j - y_j) \qquad (7)$$

where dj is the desired output of the node j, yj is the actual output and is the derivation of the activation function calculated at netj. If the node j is an internal node, then the sensitivity is defined as

$$\delta_j = f'(net_j)\sum_k \delta_k w_{jk} \qquad (8)$$

where k sums over all nodes in the layer above the node j. Update equations are derived using the chain derivation rule applied to the LMS training criterion function. Convergence can be faster if a momentum term is added and weight changes are smoothed by

$$w_{ij}(t+1) = w_{ij}(t) + \iota \delta_j x_i^{'} + \alpha\left[w_{ij}(t) - w_{ij}(t-1)\right] \qquad (9)$$

V. Repeat by Going to Step 2

The program based on the back propagation algorithm as described above, trains the network for forecasting the aircraft trajectories. This network takes input-output vector pairs during training. The network trains its weight array to minimize the selected performance measure, i.e., error using back propagation algorithm.

Our designed neural network system takes following as inputs from the user:

a) The input pattern file

b) No. of neurons in each hidden layer

c) Value of learning rate

d) Value of momentum constant

e) Error value for convergence

The output of training program is a file which contains modified weights of different connection of the network. This file is used as the input to testing program. This file also contains the values of numbers of neurons in input layer, Hidden layers, output layer, value of learning rate and momentum factor so that user is no further required to re-enter these values during testing.

3. Prediction of Aircraft Trajectory using Feed Forward neural networks

To create a database for training the neural network system as described in last section, we will be using a set of pre-recorded trajectories of aircrafts. The trained neural network system for various trajectories of aircrafts will then be used for forecasting the new one. Time is discretized so that trajectories are represented with points sampled every 15 s. A vertical trajectory will then be a set of altitudes z_0, z_1, z_2, . . . where z_0 be the altitude at $t = 0$ s and z_i corresponds to $t = 15 * i$ seconds. The neural network is of feed forward type and consists of the no. of units in input and hidden layers is n, and n be the number of inputs. The architecture of network can be shown in figure 1.

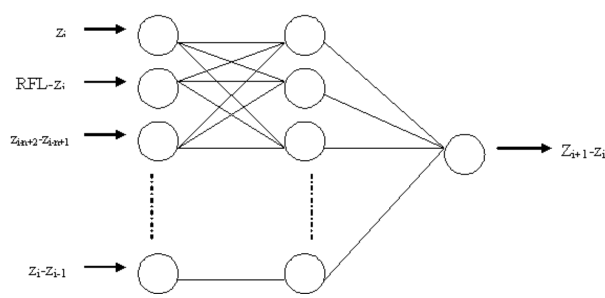

Figure 1 Showing Network Architecture

The input to the neural network is the current altitude and n past vertical speeds i. e.

z_i = Current altitude

z_{i-n+1}-z_{i-n}-…….z_i-z_{i-1} = n past vertical speeds

and the output is

$[z_{i+1} - z_i]$ = speed to predict

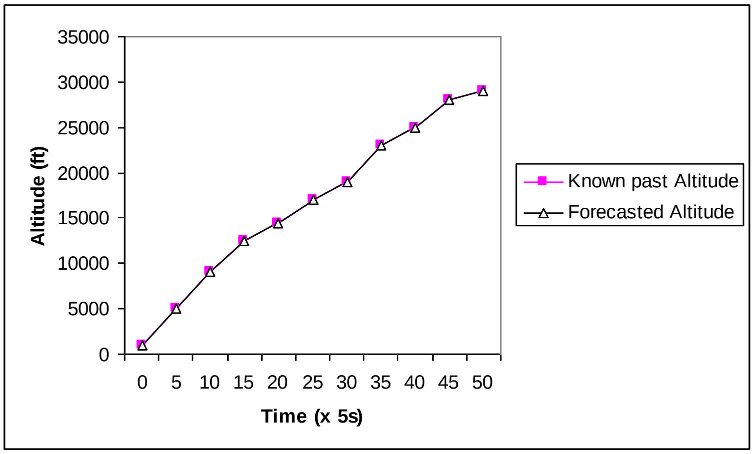

Figure 2 Arbitrary predictions for trajectories

The current altitude is the vital input as no aircraft can go high the same way at 4000 ft and 5000 ft. Also an aircraft can not climb the same way when it is 3000 ft away from its final altitude and when it is only 300 ft away, we must also provide fundamental information: the Requested Flight Level (RFL). The data is given with the quantity $RFL-z_{i,}$ where z_i represents the current altitude. The Requested Flight Level can not get directly because the aircraft has to intercept this flight level smoothly. In this way knowing the difference $RFL-z_i$ pilot can decide when to decrease the vertical speed. Now, the neural networks architecture chosen is given on figure 1 and the input patterns used are composed of:

$[z_i]$, $[RFL - z_i]$, $[z_{i-n+1} - z_{i-n}, \ldots, z_i - z_{i-1}]$

and the output is still

$[z_{i+1} - z_i]$

An algorithm that can forecast positions in a too far future is not efficient, as prediction can be changed when modifications occur. The only way to do this is to include the real points in the patterns in order to anticipate further positions with a slight delay δt.

The method is called sliding window and can be explained as following:

1. First, use n known speeds to make a prediction at $\delta t = 10 * \delta i$. Using $z_{i-n+1}, \ldots,$ z_i that represent the sliding window, the altitude $z_{i+\delta i}$ can be forecasted using the Sliding window method
2. In order to predict $z_{i+\delta i+1}$ the next known point z_{i+1} is included and $z_{i-n+2}, \ldots,$ z_{i+1} is used as the starting data (i.e the sliding window is moved one step forward to forecast the next point).
3. These steps are iterated until the aircraft has reached its *RFL*.

As it is reactive, real initial points must be provided in order to get the prediction.

Here, the method sliding windows has been applied with a network trained with 200 trajectories from the learning base and used for prediction with these 200 trajectories and with 50 non-learnt flights.

Table 1: Showing Mean, Max errors and Standard deviation for Sliding Window method

S. No.	δt	Mean error	Max error	Std Deviation
1	1 min	12 ft	290 ft	10 ft
2	2 min	152 ft	2450 ft	80 ft
3	3 min	256 ft	3245 ft	140 ft
4	4 min	346 ft	3500 ft	180 ft
5	5 min	484 ft	3555 ft	210 ft

Table 1 shows the corresponding results while figures 3 can be the example of prediction.

The smaller δt , the best the prediction. If this parameter is small the method is more reactive and therefore is able to adapt rapidly to the changes occurring in the trajectory. Even if the sliding window is not used (last line in the table) prediction is not bad. Furthermore, networks can adapt to non-learnt trajectories.

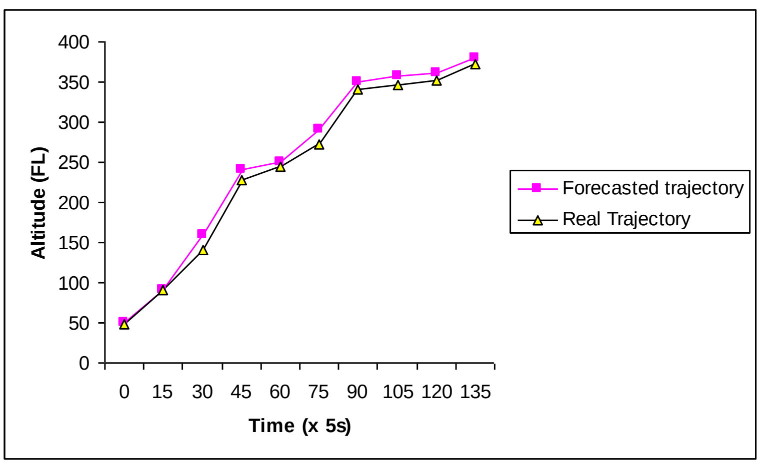

Figure 3: Showing Real trajectory with corresponding forecasted trajectory

Moreover, if the real trajectories are available while the aircraft flies, it is possible to improve precision by using past known vertical speeds and then we can only give a reactive prediction with the Sliding Window method.

4. Conclusion

Gradient descent feed forward neural networks method shows more efficient results. Moreover, they outperform the techniques currently used in the operational systems. Its application is simple and requires very less data. Neural networks can be used before and during the flight. In future, some experiments can be performed using trajectories in real time in an operational context.

References

[1] Christian Leclaire. *THAALES, Strange Trajectory Prediction, version 3.* CENA, 1994

[2] Christian Leclaire. *Technical reports on trajectory prediction technics and their evaluation.* Centre d'Etudes de la Navigation A´erienne, 1997.

[3] Thierry Mandon. *CAT: Computation of Aircraft Trajectories, version 2.* CENA, 1993.

[4] Christophe Bontemps. *Pr´evision stochastique de trajectoires : proc´edures param´etriques et nonparam ´etriques, M´emoire de DEA Informatique Fondamentale et Parall´elisme.* ENAC, 1997.