THALES Project No. 65/1311

Systematic Selection of Parameters in the development of Feedforward Artificial Neural Network Models through Conventional and Intelligent Algorithms

Research Team

G.-C. Vosniakos, T. Giannakakis, A. Krimpenis, P. G. Benardos

National Technical University of Athens, School of Mechanical Engineering,
Manufacturing Technology Division

1. Introduction

Feedforward artificial neural networks (ANNs) are currently used in a variety of applications (Figure 1) with great success. The reason behind this widespread adoption can be found in two very important abilities that they exhibit. ANNs can be trained to learn through examples (memorization ability) and can respond to cases that are similar but not identical to the ones that they have been trained with (generalization ability).

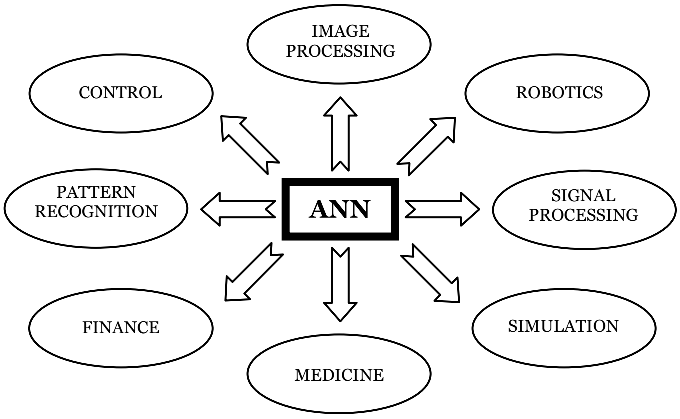
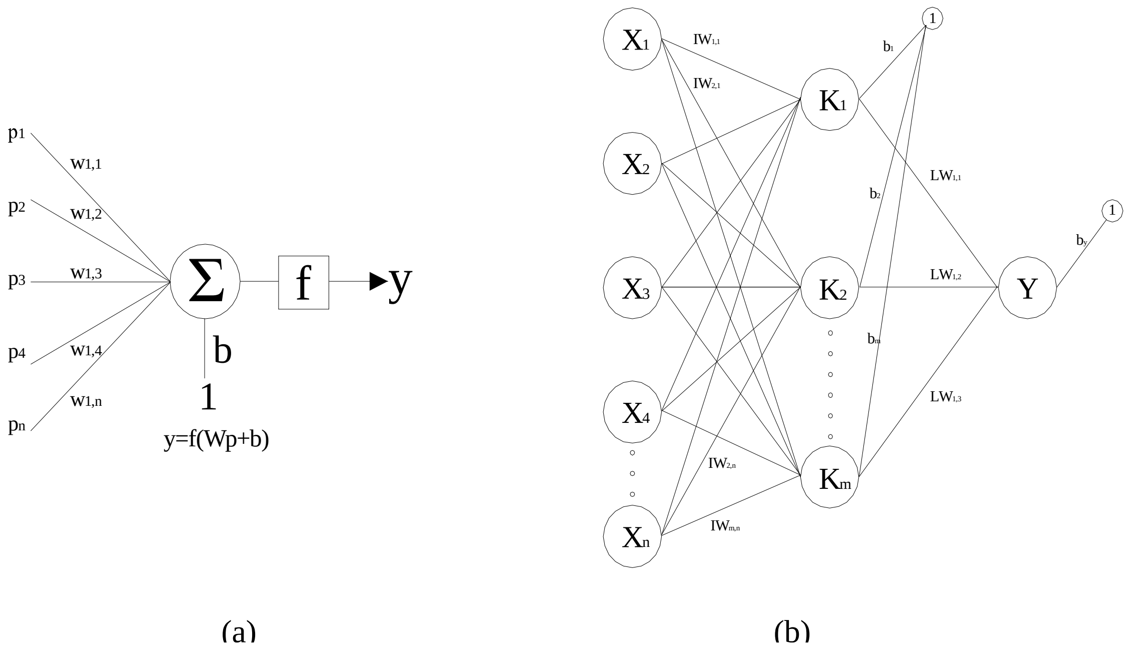


Figure 1. Applications of ANNs

The building block of a feedforward ANN is called a neuron, its mathematical model shown in Figure 2a. Each neuron, receives input in the form of weighted signals (where p is the input signal matrix and W is the weight coefficient matrix), sums them along with a bias term and applies a function f, called activation function (usually non-linear), to determine its own output signal, denoted by y.

A typical feedforward ANN is composed of several such neurons, which are arranged in layers as depicted in Figure 2b. The mathematical notation used is also given.



Figures 2. Mathematical model of (a) a typical neuron and (b) typical feedforward ANN

x_i: output signal of the i-th neuron in the input layer

k_j: output signal of the j-th neuron in the hidden layer

y: ANN's output signal

$IW_{j,i}$: weight coefficient between the i-th input neuron and the j-th hidden neuron

b_j: bias of the j-th hidden neuron

$LW_{1,j}$: weight coefficient between the j-th hidden neuron and the output neuron

b_y: bias of the output neuron

tansig(x): hyperbolic tangent function

i=1,2,…,n and **j=1,2,…,m**

The development of a feedforward ANN model involves several stages, from gathering the necessary data to creating a satisfactory model (Figure 3). The most important aspect of this process is the selection of certain parameters that are crucial for the model's performance, notably the number of hidden layers and neurons. Since there is no theoretical method to determine the appropriate architecture, a trial-and-error repetitive procedure is involved that is both time-consuming and with uncertain results. The outcome is mainly based on the experience of the researcher regarding ANNs and the studied phenomenon.

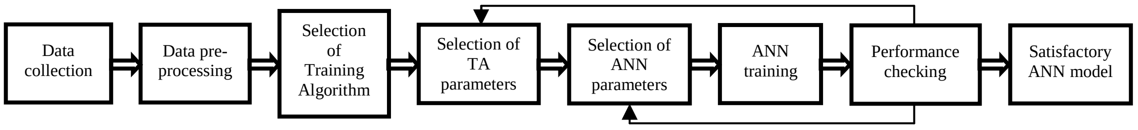


Figure 3. Development of an ANN model

The present work attempts to deal with the above problem by developing a systematic way to select such parameters. The emphasis is placed on two phases of ANN model building, namely the initialization of the network's weights and the determination of the most suitable architecture.

2. Initialization of weight coefficients

Every training procedure starts by initializing the weight coefficients, i.e. by assigning values to them. The training's goal is to find the weight values that minimize the network's error function. Since the initial values of the weights define the starting point of the training algorithm on the error function, they affect both the training speed and the achieved training error. This depends on whether this point is close to the global minimum or located in an area with many local minima. The most common methods used to initialize the weights are either to randomly select values from a predefined value field (usually centered around zero) or to use a statistical distribution (usually the Gaussian or uniform distribution). Pre-processing of the training data (normalisation, scaling etc) is also used in conjunction with these techniques.

2.1 The approach

The approach adopted is a combination of analytical and random calculation of weight values. Referring to Figure 3, the network's output is calculated below:

$$y = LW_{1,1} \cdot k_1 + LW_{1,2} \cdot k_2 + \mathbf{...} + LW_{1,m} \cdot k_m + b_y \qquad (1)$$

The output of each hidden neuron is in turn given by the following equation:

$$k_j = \tan sig(IW_{j,1} \cdot x_1 + IW_{j,2} \cdot x_2 + ... + IW_{j,n} \cdot x_n + b_j) \qquad (2)$$

If a multiple linear regression is performed on the training data then the resulting analytical model would be:

$$y = a_0 + a_1 \cdot x_1 + a_2 \cdot x_2 + ... + a_n \cdot x_n = \sum_{i=1}^{n} a_i \cdot x_i + a_0 \qquad (3)$$

Comparing equations 2 and 3, it is concluded that the argument of the hyperbolic tangent function can be replaced by the analytical model of the multiple linear regression. This is accomplished if:

$$b_{kj} = a_0 \qquad (4)$$

$$IW_{j,i} = a_i \qquad (5)$$

The remaining weights between the hidden and the output layer and the respective bias are initialized randomly so that the starting point of the training algorithm is slightly different each time the training is repeated.

2.2 Results

The approach was tested by comparing its results to the Nguyen-Widrow method. Data originating from a bar turning process were used to develop an ANN model and the number of required epochs and achieved training error (mean squared error – MSE) were examined. Three different architectures were investigated, namely 5x10x1, 5x6x1, 5x3x1, and the training results are given in Table 1.

	5x10x1		**5x6x1**		**5x3x1**		
Training no.	**Epochs**	**MSE Training Error**	**Epochs**	**MSE Training Error**	**Epochs**	**MSE Training Error**	**Initialization type**
1	1466	1,43E-25	2325	1,15E-29	5000	1,26E-06	N-W
2	734	1,54E-28	3441	2,07E-28	5000	1,60E-06	N-W
3	670	1,78E-29	10000	2,45E-07	1675	2,69E-06	N-W
4	618	2,24E-25	9377	1,95E-26	5000	6,90E-07	N-W
5	1753	8,30E-26	1397	3,79E-24	5000	1,06E-06	N-W
6	765	7,63E-28	10000	3,91E-08	5000	8,38E-07	N-W
7	983	5,90E-24	2565	1,37E-26	612	8,10E-07	N-W
8	2155	1,11E-27	2612	1,92E-28	5000	7,69E-07	N-W
9	256	1,27E-31	1701	6,85E-28	5000	9,13E-07	N-W
10	1139	4,74E-24	10000	3,23E-07	5000	1,60E-06	N-W
		1,11E-24		**6,08E-08**		**1,22E-06**	

	5x10x1		**5x6x1**		**5x3x1**		
Training no.	**Epochs**	**MSE Training Error**	**Epochs**	**MSE Training Error**	**Epochs**	**MSE Training Error**	**Initialization type**
1	915	5,61E-31	2351	4,97E-27	6393	1,03E-06	MLR
2	686	1,72E-31	1473	1,25E-30	6576	1,03E-06	MLR
3	750	1,67E-28	2294	2,81E-29	6510	1,03E-06	MLR
4	1328	8,47E-26	1666	3,74E-30	10000	6,94E-07	MLR
5	1004	5,54E-31	957	3,73E-29	10000	8,28E-07	MLR
6	985	1,86E-26	2147	1,09E-30	6752	1,03E-06	MLR
7	1032	2,12E-26	926	6,32E-28	6990	1,03E-06	MLR
8	899	2,52E-28	996	7,54E-31	10000	8,28E-07	MLR
9	903	1,04E-28	1926	2,50E-30	10000	8,28E-07	MLR
10	1860	3,71E-31	1019	8,82E-29	7030	1,03E-06	MLR
		1,25E-26		**5,76E-28**		**9,34E-07**	

Table 1. Initialization method results

It is observed that there is improvement in both of the examined parameters, which is proportionately higher to the complexity of the architecture.

3. Determination of ANN’s architecture

An ANN’s architecture is directly related to the complexity of the solution space that it represents. A network that is fairly simple might not be able to learn the interactions underlying the training data, while a very complex network will memorize them to such extent that it will no longer be able to respond to unknown data. Obtaining the right architecture is the most crucial stage in the development of an ANN model and given that there is no theory as to what this architecture is or how to obtain it, it is also one of the most difficult stages to perform. Current practice involves a trial-and-error approach, but there are a lot of research efforts involving the use of evolution algorithms as well as constructive/deconstructive analytical techniques that try to address this problem.

3.1 The approach

If the described problem is viewed as a problem of multi-parametric optimization, then a genetic algorithm can be used. The aim is to find the appropriate architecture, i.e. the number of hidden layers and the number of neurons in each one of them, which results in an ANN model with good performance. In order to satisfy this, criteria that quantify the performance of the model are developed and are consequently integrated in the objective function to be minimized. These criteria are:

i. Training error criterion

$$E_{training} = \frac{\sum_{i=1}^{n} \left| \frac{Y_{o_i} - Y_i}{Y_{o_i}} \right|}{n}$$

, where **E_{training}:** training error, Y_{o_i} : target value of the i-th training data vector, Y_i : ANN's response to the i-th training data vector and **n:** number of training data

ii. Generalization error criterion

$$E_{generalization} = \frac{\sum_{i=1}^{n} \left| \frac{Y_{o_i} - Y_i}{Y_{o_i}} \right|}{n}$$

, where $\mathbf{E_{generalization}}$**:** generalization error, Y_{o_i} : target value of the i-th testing data vector, Y_i : ANN's predicted value for the i-th testing data vector and **n:** number of testing data

iii. Feedforward architecture criterion

$$\mathbf{FFAC} = \begin{cases} \mathbf{1} \text{ , 1 hidden layer and } m \le 10 \\ \mathbf{1+(m-10)*0.1} \text{ , 1 hidden layer and } m > 10 \\ \mathbf{2} \text{ , 2 hidden layers and } m \le 10 \text{ and } n \le 10 \\ \mathbf{2+(m-10)*0.1} \text{ , 2 hidden layers and } m > 10 \text{ and } n \le 10 \\ \mathbf{2+(m-10)*0.1+(n-10)*0.2} \text{ , 2 hidden layers and } m > 10 \text{ and } n > 10 \end{cases}$$

, where **m** and **n:** number of neurons in the 1st and 2nd hidden layer respectively

iv. Training speed criterion

$$trspeed = \begin{cases} 1.5 \ , \ epochs < 10 \\ 1 \ , \ epochs > 10 \end{cases}$$

v. Solution space consistency criterion

$$solspc = 1 + x * 0.33 + y$$

, where **x:** number of test cases that the absolute value of the relative error is in the interval [15,25] and **y:** number of test cases that the absolute value of the relative error is in the interval (25, ∞)

3.2 Results

Using the same data as for the initialization method testing, the developed method was compared to the results of an experienced researcher that followed the trial-and-error approach. The model achieved by an experienced human analyst was 5x3x1.

The best objective function value versus the number of generations is shown in Figure 4 and the number of neurons in each layer is given in Table 2.

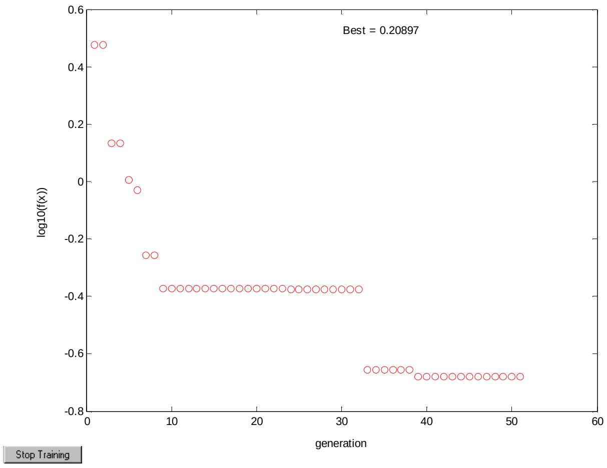


Figure 4. Best objective function value history

1st hidden layer	2nd hidden layer
3	0
1	0
3	0
3	1
3	5
3	0
3	0
2	4
3	4
3	0
3	0
11	0
3	0
10	16
3	2
3	0
3	0
3	0
1	0
7	4
19	0
1	0
3	0
3	0
19	0

Table 2. Number of neurons in each hidden layer

As can be seen from the above table, the developed methodology performs as well as a human expert, but at the same time it offers advantages such as no required experience, shorter development time and systematic selection of the network's parameters.

4. Conclusions

By using the described methodologies, the development of an ANN model is facilitated and, more importantly, it is carried out following a systematic procedure, rather than a repetitive trial-and-error procedure with uncertain results. In both fields (weight initialization and architecture determination), the results show an improvement over current practices. Furthermore, in the latter case, the focus is primarily on the generalization performance of the ANN and network size, which guarantee accurate and consistent model predictions.

Publications

1. "Initialisation improvement in engineering feedforward ANN models", 13th European Symposium on Artificial Neural Networks, 27-29 April 2005, Bruges, Belgium.
2. "Optimising feedforward artificial neural network architecture", Engineering Applications of Artificial Intelligence, submitted for publication.