The Effect Of Number Of Training Samples For Feed Forward Neural Network

Luma. N. M. Tawfiq & Alaa K. J. AL-Mosawi

Department of Mathematics,College of Education, Ibn Al-Haitham, Baghdad University.

Abstract

In this paper we study the effect of the number of training samples for feed forward neural networks (FFNN) which is necessary for training process of feed forward neural network. Also we design 5 FFNN's and train 41 FFNN's which illustrate how good the training samples represent the actual function for FFNN.

1.Introduction

Artificial Neural Networks (Ann) are relatively crude electronic models based on the neural structure of the brain. The brain basically learns from experience. It is a natural proof for some problems, that are beyond the scope of current computers, are indeed solvable by small energy efficient packages. This brain modeling also promises a less technical way to develop machine solutions. This new approach of computing also provides more graceful degradation during system overload than its more traditional counterparts.

Now, advances in biological research promise an initial understanding of the natural thinking mechanism. This research shows the brains store information as patterns. Some of these patterns are very complicated and allow us the ability to recognize individual faces from many different angles. This process of storing information as patterns, utilizing those patterns, and then solving problems encompasses a new field in computing. This field, as mentioned before, does not utilize traditional programming but involves the creation of massively parallel networks and the training of those networks to solve specific problems. This field also utilizes words very different from traditional computing, words like behave, react, self-organize, learn, generalize, and forget. [1]

2. Artificial Neurons and How They Work

The fundamental processing element of artificial neural network (Ann) is a neuron. A neuron is an information-processing unit that is fundamental to the operation of a neural networks, Figure (1) shows the model of a neuron, which forms the basis for Ann's. The artificial neurons that we use to built our neural networks are truly primitive in comparison to those found in the brain. An artificial neuron has several inputs but only one output. Here we identify three basic elements of the neuronal model [2], [3]:

1. *Synapses* or connecting links, each is characterized by a weight or strength of its own. Specifically, a signal x_j at input of synapse j connected to a neuron is multiplied by the synaptic weight w_j.
2. An *activation function* for limiting the amplitude of the output of a neuron.
3. The model of a neuron also includes an external *bias*, denoted by b, which has the effect of increasing or lowering the net input of the activation function.

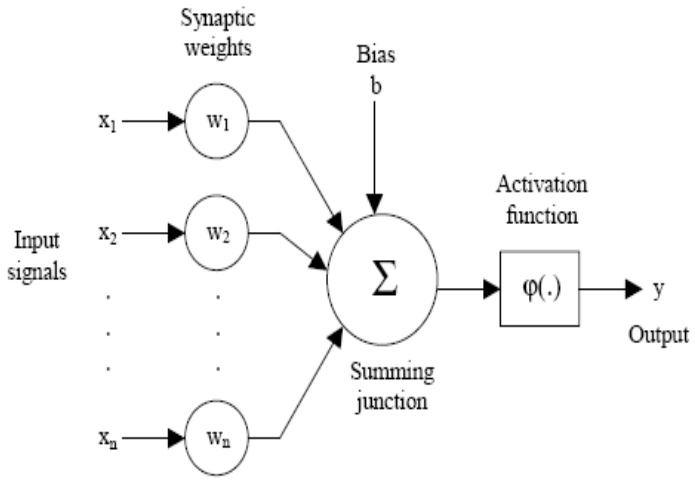


Figure 1: *Nonlinear model of a neuron*

In mathematical terms, we may describe a neuron by:

$$y = \varphi\left(\sum_{j=1}^{N} w_j x_j + b\right) \quad \ldots\ldots(1)$$

Where $x_1, \ldots x_n$ are the input signals, $w_1, \ldots, w_n$ are the synaptic weights of the neuron, b is the bias, φ is the activation function and y is the output signal of the neuron.

1.4. Training of Ann's

Basically, training is the process of determining the weights which are the key elements of an Ann. The knowledge learned by a network is stored in the nodes in the form

of weights and node biases. In this paper network training that the desired response of the network (target value) for each input pattern (example) is always available. The training input data is in the form of vectors of input variables or training patterns. Corresponding to each element in an input vector there is an input node in the network input layer. Hence the number of input nodes is equal to the dimension of input vectors. The total available data is usually divided into a training set (in-sample data) and a test set (out-of-sample). The training set is used for estimating the weights while the test set is used for measuring the generalization ability of the network.

The training process is usually as follows. First, examples of the training set are entered into the input nodes. The activation values of the input nodes are weighted and accumulated at each node in the first hidden layer. The total is then transformed by an activation function into the node's activation value. It in turn becomes an input into the nodes in the next layer, until eventually the output activation values are found. The training algorithm is used to find the weights that minimize some overall error measure such as the sum of squared errors (SSE) or mean squared errors (MSE). Hence the network training is actually an unconstrained nonlinear minimization problem. [4]

4. Feed forward Neural Network

Feed-Forward Neural Network (FFNN) has a layered structure. Each layer consists of units which receive their input from units from a layer directly below and send their output to units in a layer directly above the unit. There are no connections within a layer. The N_i inputs are fed into the first hidden layer of $N_{h,1}$ hidden units. The input units are merely 'fan-out' units; no processing takes place in these units. The activation of a hidden unit is a function F_i of the weighted inputs plus a bias, as given in equation (1). The output of the hidden units is distributed over the next layer of $N_{h,2}$ hidden units, until the last layer of hidden units, of which the outputs are fed into a layer of N_o output units.

Although back-propagation can be applied to networks with any number of layers. In most applications a feed-forward network with a single layer of hidden units is used with a sigmoid activation function (bounded, monotonically increasing and differentiable, where these sigmoid satisfy the boundary condition: $\lim_{x\to\infty}\sigma(x)=1$, $\lim_{x\to-\infty}\sigma(x)=0$) [5], [6] for the units.

This activation function also called transfer function or mathematically basis function. In this thesis we depend the results in [7] for choosing most transfer function, which is '*tansig*'.

5. An example

A feed-forward network can be used to approximate a function from data. Suppose we have a system (for example a chemical process or a financial market) of which we want to know the characteristics. The input of the system is given by the two-dimensional vector (x,y) and the output is given by the one-dimensional vector z. We want to estimate the relationship z = f(x,y) from 80 data {(x,y), z} as depicted in figure (2) (top left). A FFNN was programmed with two input units, 10 hidden units with '*logsig*' activation function and an output unit with a linear activation function. The network weights are initialized to small values and the network is trained for 5000 training iterations with the back-propagation (*gradient descent*) training rule. The relationship between (x,y) and z as represented by the network is shown in figure (2) (top right), while the function which generated the training samples is given in figure (2) (bottom left). The approximation error is depicted in figure (2) (bottom right). We see that the error is higher at the edges of the region within which the training samples were generated. The network is considerably better at interpolation than extrapolation.

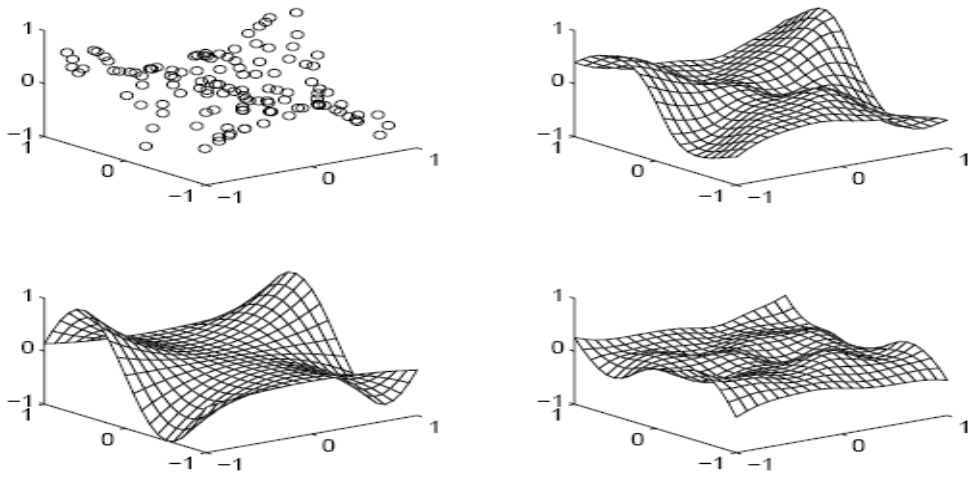


Figure 2: Example of function approximation with a FFNN. Top left: The original training samples; Top right: The approximation with the network; Bottom left: The function which generated the training samples; Bottom right: The error in the approximation.

6.How good are multi-layer feed-forward networks?

From the example shown in figure (2) it is clear that the approximation of the network is not perfect. The resulting approximation error is influenced by:

1. The training algorithm and number of iterations: This determines how good the error on the training set is minimized.
2. The number of training samples: This determines how good the training samples represent the actual function.
3. The number of hidden units: This determines the "expressive power" of the network. For "smooth" functions only a few number of hidden units are needed (2N+1) [7], for wildly fluctuating functions more hidden units will be needed.

In this paper we particularly address the effect of the number of training samples.

We first have to define an adequate error measure. All neural network training algorithms try to minimize the error of the set of training samples which are available for training the network. The average error per training sample is defined as the training error rate:

$$E_{learning} = (1/P_{learning}) \sum_{p=1}^{P_{learning}} E^p$$

in which E^p is the difference between the desired output value and the actual network output for the training samples:

$$E^p = (1/2) \sum_{o=1}^{n} (d_o^p - y_o^p)^2$$

This is the error which is measurable during the training process.

It is obvious that the actual error of the network will differ from the error at the locations of the training samples. The difference between the desired output value and the actual network output should be integrated over the entire input domain to give a more realistic error measure.

This integral can be estimated if we have a large set of samples: the test set. We now define the test error rate as the average error of the test set:

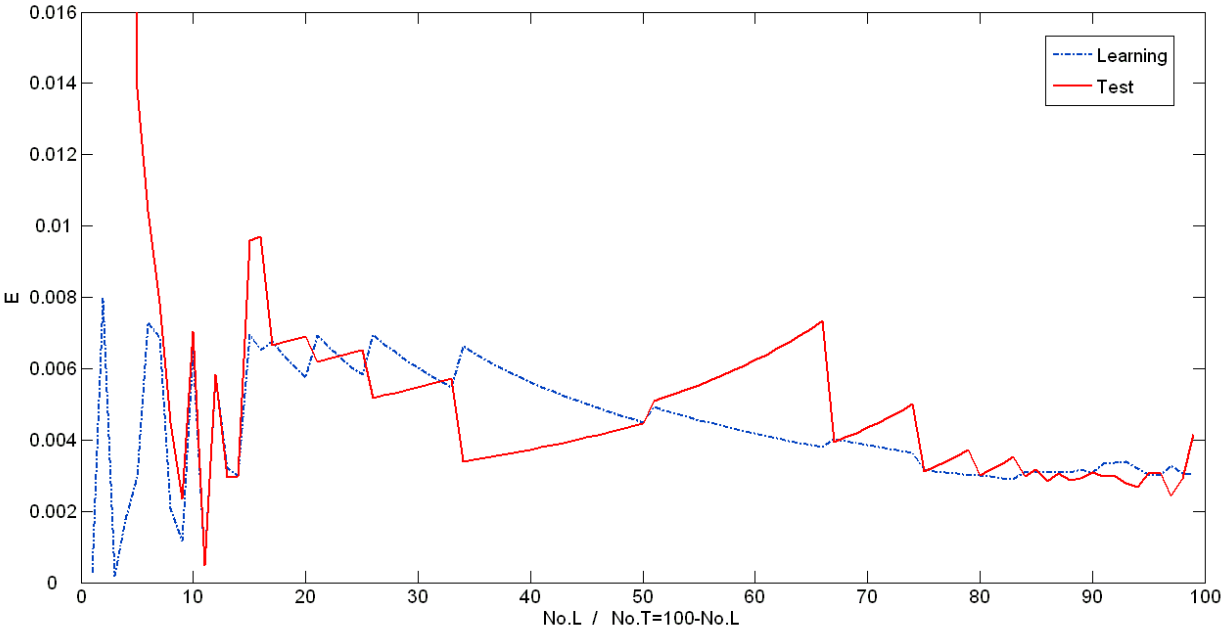
$$E_{test} = (1/P_{test}) \sum_{p=1}^{P_{test}} E^p \ .$$

In the following subsections we will see how these error measures depend on the training set size and the number of hidden units.

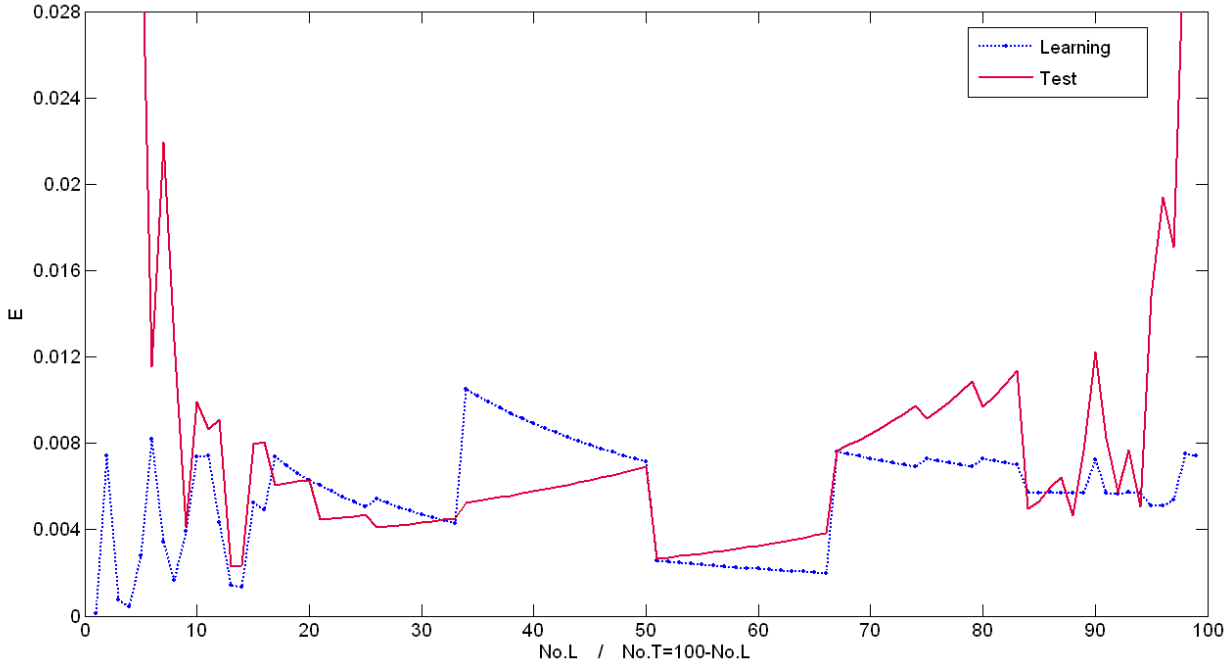
7. The effect of the number of training samples

A simple problems are used as an example: a function $y_i = f_i(x)$, i=1,2,3,4, has to be approximated with a FFNN. A neural network is created with an input, 3 hidden units with '*tansig*' activation function and a linear output unit in the first three problems and two input units, 5 hidden units with '*tansig*' activation function and a linear output unit in the fourth problem. Suppose we have L number of training samples variant from 1 to 100. The networks are trained with these samples. Training is stopped when the error does not decrease anymore and the number of testing sample is 100–L. The training samples and the testing sample of the network are shown in the same figure. We see that in the interval [1,20] and [90, 100] $E_{training}$ is small (the network output goes perfectly through the training samples) but E_{test} is large: the test error of the network is large.

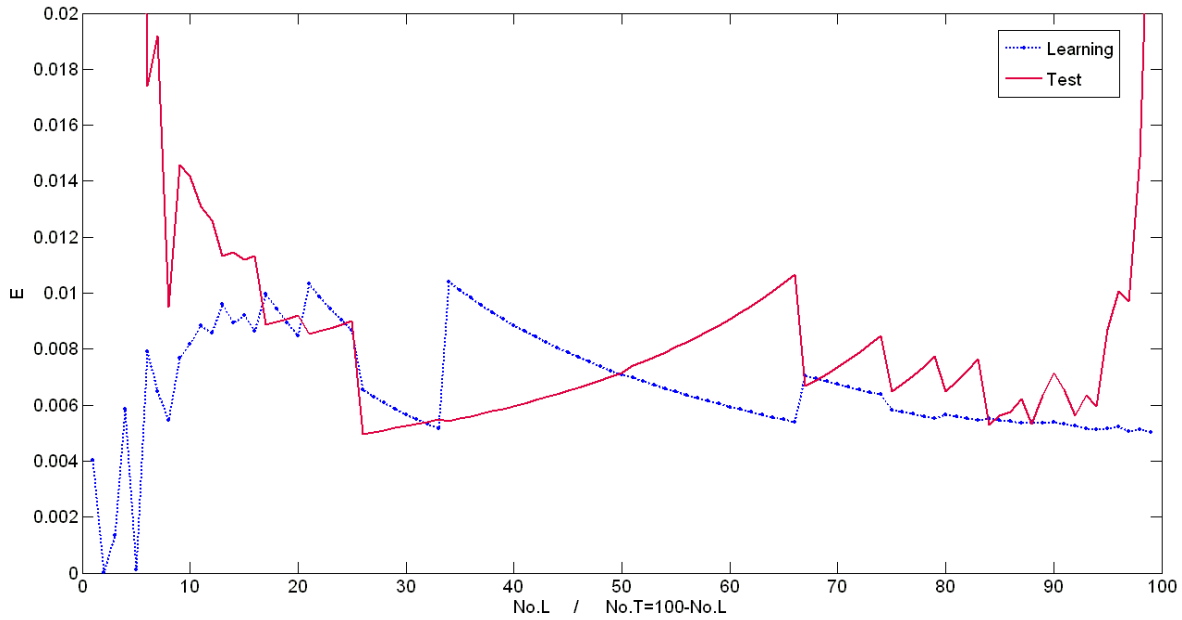
These experiment was carried out with other training set sizes, where for each training set size the experiment was repeated 10 times. The average training and test error rates as a function of the training and test set size are given in figure (3). Note that the training error increases with an increasing training set size, and the test error decreases with increasing training set size. A low training error on the (small) training set is no guarantee for a good network performance! With increasing number of training samples the two error rates converge to the same value. This value depends on the representational power of the network: given the optimal weights, how good is the approximation. This error depends on the number of hidden units [8] and the activation function. If the training error rate does not converge to the test error rate the training procedure has not found a global minimum.



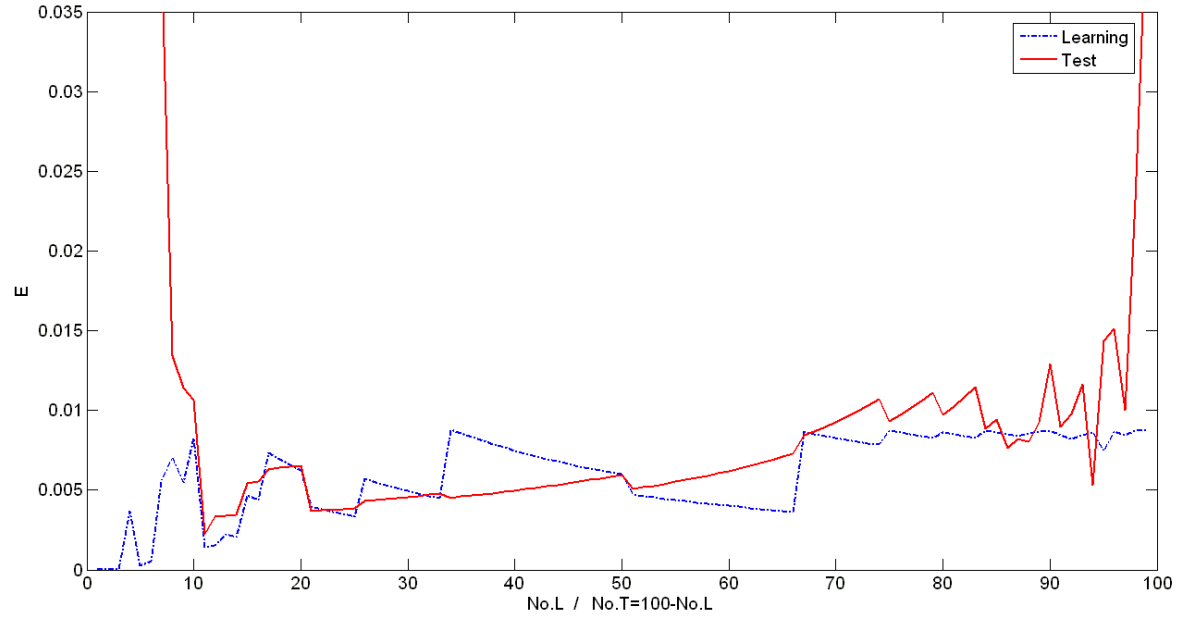
$Y_1 = f_1(x) = sin(x)$



$Y_2 = f_2(x) = 2x^5 + 6x^3 - 9x^2 + 1$



$$Y_3 = f_3(x) = 3x\,(x - 0.6\,)\,(x + 1.17)$$



$$Y_4 = f_4(x) = (\,x_1 - x_2\,)^3\,18\,x_1\,(x_2 - x_1)7x_2$$

Figure 1.4: Effect of the training set size on the error rate. The average error rate and the average test error rate as a function of the number of training samples with L training samples and 100 – L testing samples for 4 problems

References

[1] B.C.Csáji, "**Approximation with Artificial Neural Networks**", MSC thesis, Faculty of Sciences, Eötvös Loránd University, Hungary, 2001.

[2] G.P.Jaya Prakash and TRBstaff Representative,"**Use of Artificial Neural Networks in Geomechanical and Pavement System**", Transportation Research Circular, Number E-co12, December 1999.

[3] N.Alldrin, A.Smith and D.Turnbull, "**A Three – Unit Networks is All You Need to Discover Females**", D.Simonl Nerocomputing, 2001.

[4] A.Pinkus, "**Approximation theory of the MLP model in neural networks**", Acta Numerica , pp.143-195,1999.

[5] G.Zhang, B.Eddy patuwoand Y.Hu, "**Forecasting with Artificial Neural Networks: The State of the Art**", International Journal of Fore casting No.14,pp.35-62,1998.

[6] E.W.Dijkstra, "**Approximation with Artificial Neural Networks**", MSc thesis, Faculty of Mathematics and Computing Science Eindhoven University of Technology, The Netherlands, 2001.

[7] Q.Hatim, "**Comparison of Ridge and Radial Basis Functions Neural Networks for Interpolation Problems**", Msc Thesis, Baghdad University, College of Education - Ibn Al-Haitham, 2007.

[8] L.N.M.Tawfiq and Q.H.Eqhaar, "**On Multilayer Neural Networks and Its Application for Approximation Problem**", 3rd scientific conference of the College of Science, University of Baghdad. 24 to 26 March 2009.

تاثير عدد عينات التدريب في الشبكات العصبية ذات التغذية التقدمية

د. لمى ناجي محمد توفيق و علاء كامل جابر

قسم الرياضيات – كلية التربية – ابن الهيثم – جامعة بغداد

المستخلص

في هذا البحث درسنا تاثير عدد عينات التدريب في الشبكات العصبية ذات التغذية التقدمية والتي تعتبر ضرورية في عملية التدريب لهذا النوع من الشبكات ايضا صممنا 5 شبكات عصبية ذات تغذية تقدمية ودربنا 41 شبكة عصبية ذات تغذية تقدمية توضح كيف ان الاختبار الجيد لعدد عينات التدريب يؤدي الى تقريب او تمثيل جيد للدالة الحقيقية.