Neural Networks for cost estimating in project management

By

Gregor Sandhaus

Diplom-Wirtschaftsingenieur

A Thesis Submitted to the University of Wales in the Fulfilment of the Requirements for the Degree

of

Doctor of Philosophy

European Business Management School
University of Wales

July 1998

Certification of Research and Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Candidate

Date 13.10.98

Summary

This thesis considers the application of neural networks in cost estimating in project management and whether they lead to more accurate estimates. It strikes two areas of research, namely neural networks and project management; an introductory chapter on both subjects is included. The statistical problem of parametric cost estimating is described and an explanation of the general principles is given. The Multi-Layer Perceptron with the Backpropagation learning algorithm is determined to be the most appropriate network and a selection of available software programs is reviewed.

A Multi-Layer Perceptron neural model is used to determine one of the most important cost estimating relationships of the PRICE model. A comparison of the outputs of the neural network and the PRICE model shows that the Backpropagation algorithm is able to find the underlying estimating relationships used by PRICE.

To investigate whether other underlying functions can be found with artificial intelligence methods, other input parameters are selected and the costs generated by the PRICE model and by the neural network are compared with each other. Further experiments were undertaken in order to improve the performance of the neural network. The neural networks were applied to real data. and their output compared with the PRICE model. The processes of achieving better results are analogous to those used for the artificial data. A neural network was created which performs better than the PRICE model in terms of the accuracy of the estimates produced.

The results are discussed and the collection of significant and accurate information and then deciding on which type of network is the best network to be used are identified as the major problems in the application of artificial intelligence for cost estimation in project management. The limitations and restrictions of the implementation of neural networks are examined and the scope and topics of further research are suggested.

Acknowledgements

I would like to thank my supervisor, Dr. Rhys G. Williams, for his valuable supervision and discussions during the course of the work. Without his constant encouragement, it would be impossible to complete this study.

Many thanks go to Mr. Huebner, Claus Petters, Stefan Maetschke, Heinz Mehler, Mr. Bettinger and Prof. Dr. Bernd Blümel for their enlightening suggestions and for providing the initial sources of information.

Last but not least thank you to Prof. Dr. Jörg Liese and the Deutsche Akkademische Austauschdienst (DAAD) who gave me the opportunity to thoroughly focus on this research.

BLANK PAGE IN ORIGINAL

Table of Contents

1 General introduction and outline of the thesis

Project management depends on achieving the goal of the desired outcomes or product of a project within deadlines and within resources constraints.

Cost estimates should be undertaken in order to evaluate whether the expected inputs and outputs are acceptable. If a company decides to apply formal methods for cost estimation within the organisational structure there are several methods they can choose from.

Chapter 3 reviews the literature on cost estimating in project management. A distinction is made between different methods of cost estimates. Parametric cost estimating methods are explained in more detail as these are among the most frequently applied methods and particularly as their mathematical framework provides a basis for following chapters of the thesis.

The application of cost estimating relationships (CER) in parametric estimating methods assumes a correlation between those components which have been used to establish the CER and that element which is estimated. Lockheed Martin Marietta PRICE Systems is the market leader in providing a computer aided project cost estimating tool based on the CER. PRICE uses parameters including for example the length of production run, the number of prototypes, volume, platform, production complexity, development complexity, starting date and deadline for development and production as universal cost drivers to determine the costs of a component.

When new CERs are generated a regression analysis is performed and that is based on many data samples. The major disadvantage of this regression analysis is that it is only able to find linear functions in the data set. Particularly for multi-dimensional data (e. g. several input CERs to determine the cost) finding non linear functional connections is at least very complicated if not impossible. Since

it can be assumed that many CERs are not linear to their costs it would be very interesting to develop a procedure which is able to find non linear relationships.

This is where artificial intelligence may be useful. Neural networks are able to detect interdependendencies between input and output parameters during a learning phase and then to apply this knowledge to comparable data. To understand the basic ideas of neural networks and their mathematical background chapter 2 of this thesis provides an introduction to artificial intelligence and describes the structure and learning algorithm of a selection of some typical neural networks computer software. Typical applications and their limitations in these areas are also reviewed.

Project cost data generated from the PRICE model is applied to train a neural network with this information. The object is to determine to what extent a neural network is capable of simulating an already existing mathematical model that is based on CERs.

The next section examines whether neural networks are able to represent the complete cost function of the PRICE model by applying input parameters other than complexity values. Several tests were undertaken in order to improve the performance of the neural network. Doing so lead to a closer insight into training, data normalisation and other possible influences of data representation.

The results indicate that the neural networks are quite sensitive to the data and the underlying function. Extensive statistical analysis on the data was necessary and therefore the question arises whether it is still useful to use artificial intelligence if all the statistical work cannot be avoided.

In the last test a new normalisation method is developed which does not require any statistical analysis at all and the results of the neural networks were acceptable. It still had to be shown that this kind of normalisation can also lead to good results for other data sets where the underlying function is different to the applied data base. The next task is to show that neural networks can perform better on real world data than these existing parametric cost estimating models.

With the help of real cost data provided by Daimler Benz Aerospace Airbus in Germany (DASA), the complexity value of the PRICE model was determined and then compared to an calculated complexity value of the neural net.

The neural networks performance improved within several tests which where similar to the previous chapter except that this time an extensive statistical analysis was not possible due to lack of sufficient data sets.

The new normalisation method which was applied within the last tests again lead to the best results. It can be asserted that this new normalisation method can help the neural network to perform acceptably well.

In the last chapter the discussion and conclusions of the findings throughout the thesis are included. Gathering significant information and applying the possibly best network are stressed as major problems in the use of artificial intelligence for cost estimation in project management. The limitations and restrictions of the implementation of neural networks are examined and the potentials for further research are suggested.

2 Neural Networks

2.1 Introduction

Neural network technology is a relative newcomer to many application areas. Its widespread use today is fraught with difficulties in achieving optimum performance, of monitoring their activity, and integrating them into existing solutions and operating environments. But most of all, there is a purely practical process of learning involved in understanding what neural networks are about, how they operate, and what they can do.

This chapter presents a brief overview of the principles of artificial neural networks and describes some of the most important types of networks.[1]

2.2 The Human Brain

The brain receives, processes and answers stimuli through receptors. Receptors register light, pressure, temperature etc. and code these stimuli in such a way that they can be sent by the nervous system to the brain. Most signals reach the neocortex in the brain. The elementary processing units in the brain are neurons (see Figure 1). These cells exchange electrochemical signals that can excite each other. Like any other processor they have input and output. They receive the electrochemical stimulus through dendrites from other neurons. The neuron becomes active if the electrochemical input exceeds a certain level. It sends an electrical impulse via the axon to other neurons. Between the dendrites and the neuron is a cell that either excitates or inhibits the signal. These cells are called synapses. The synapses can change its effect on the signal over the time.

---

[1] Lisboa, P. G. J.: neural networks - current applications, Chapman & Hall, 1992, London, p. XVIII

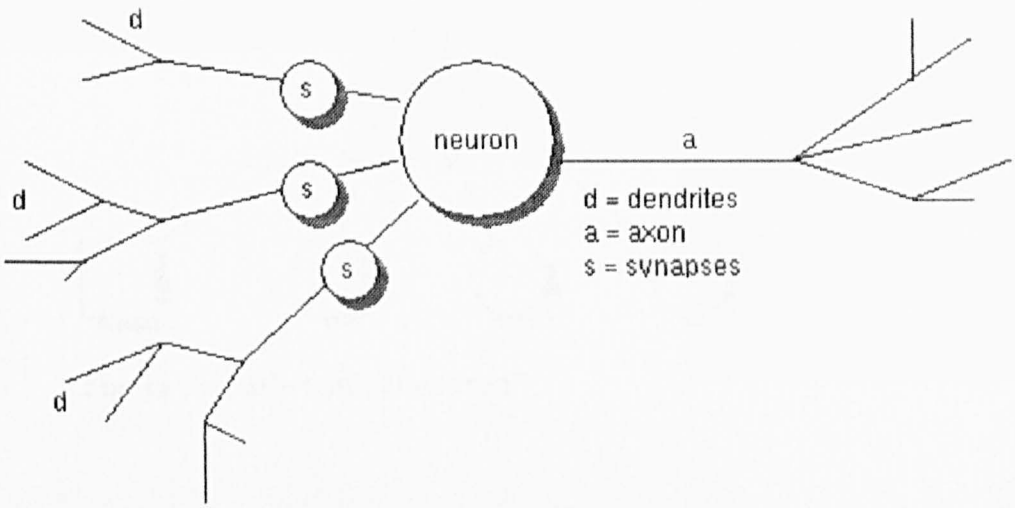


Figure 1: basic structure of the human neuron

Therefore it is essential for the learning process. In an artificial neural network the synapses is represented by a positive or negative factor.[2]

The human brain has about 10^{10} neurons and each neuron is connected with about 10,000 other neurons that results in 10^{14} nodes altogether. Putting all axons in a line they reach a length of about 500,000 km. At birth all neurons already exist although they are not connected. The connection starts with the learning process. Learning therefore describes the building up of connections between neurons. Every training strengthens the connection while missing training disintegrates the connection, that is to forget.[3]

The brain functions associatively. That means that the brain sets up group of neurons (assemblies) that cover other assemblies for example some neurons represent both assembly "lightning" and assembly "thunder". The activation of assembly " lightning " therefore activates part of the assembly "thunder" or: If I see lightning I automatically think of thunder, too. [4]

---

[2] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 11, 12

[3] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 13

[4] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 14-15

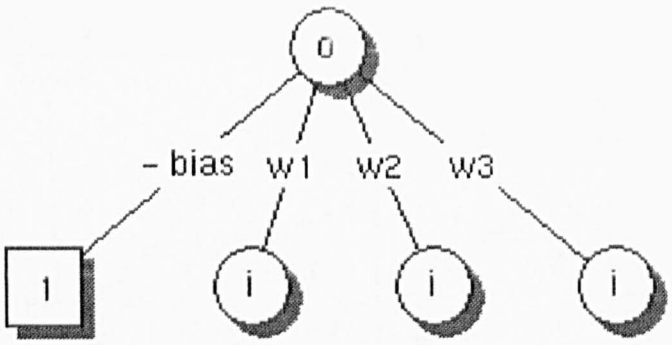


Figure 2: mathematical neuron[6]

2.3 Mathematical Description

A neuron is a processor that is either active or inactive. This is represented with the binary values of 0 and 1. Because the input values come from other neurons they are either 0 or 1, too. The strength of each synaptic junction is represented by a multiplicative factor or weight, with a positive sign for excitatory connections and negative otherwise. If the sum of incoming signals exceeds a certain threshold then the neuron becomes active (it "fires").[5] Mathematical summary:

Definition (neuron): $i_1, i_2, i_3, \ldots i_n$ are input values of 0 or 1. $w_1, w_2, w_3, \ldots w_n$ are synapses values of any number (weights) and

$net = \sum w_j * i_j$

represents the cumulative input. Then

$o = f(net-\theta)$

is the output of the neuron. The function f(x) is defined with: $f(x) = \begin{cases} 1 \text{ if } x > 0 \\ 0 \text{ if } x \leq 0 \end{cases}$.

In Figure 2 the neuron is shown graphically. f(x) represents the transfer function. Common transfer functions are the binary function, the signum function $f(x) = \begin{cases} 1 \text{ if } x > 0 \\ -1 \text{ if } x \leq 0 \end{cases}$ or the squashing function $f(x) = 1/(1 + \exp(-c*x))$. Mathematically

---

[5] Ritter, H., Martinetz, T., Schulten, K.: Neuronale Netze, Addison-Wesley, 1992, p.25

[6] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 17

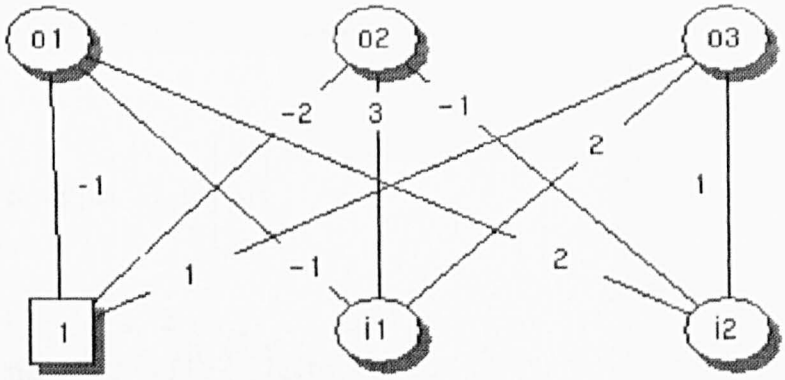


Figure 3: Example of a neural network

the threshold (bias) can be represented as fixed incoming input signal. If so then θ = 0[7].

2.4 Neural Networks

Connecting several neurons with each other creates a neural network (Figure 3). The net inputs are i_1 and i_2 and the outputs are o_1, o_2 and o_3. The weights are shown on each connection. For every input value (i_1, i_2) one output value exists (o_1, o_2, o_3). Therefore this neural network can be represented by the following logical function:

$o = f(net-\theta)$

$o_1 = f(net1 - 0)$

$o_2 = f(net2 - 0)$

$o_3 = f(net3 - 0)$

$net = \sum w_j * i_j$

$net1 = -1*i_1 + 2*i_2 - 1$

$net2 = 3*i_1 - 1*i_2 - 2$

$net3 = 2*i_1 + i_2 + 1$

f = binary function, θ = 0, because Bias is represented as fixed incoming input signal)

[7] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 15-18

The same equations expressed in vector terms is:

$$o = f(net - \theta) = \begin{pmatrix} o_1 \\ o_2 \\ o_3 \end{pmatrix}$$

$$net = \begin{pmatrix} -1 & 2 \\ 3 & -1 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} i_1 \\ i_2 \end{pmatrix}; \theta = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

A single-layer neural network can be defined by:

Definition (single-layer neural network): A single-layer neural network can be represented by a n*m matrix W with its elements of real numbers and a transfer function f. Every input vector i is related to one output vector o according to the regulation:

$o = f(net - \theta)$ (θ = hard-limited vector)
$net = W * i$

In a graphically represented neural network every input can be connected with every output. The matrix elements (weights) are added to the line. The threshold is represented by a bias.[8]

In a multi-layer network there is at least one hidden layer (Figure 4). Its excitation levels are set by the externally imposed input pattern. These levels are inputs for the following layer. The mathematical expression for a hidden layer corresponds to a single-layer neural network:

$h = f(net1 - \theta_1)$
$net1 = W_1 * i$
(h = hidden layer, i = input vector, θ_1 = threshold vector, W_1 = matrix of weights of the first level).

---

[8] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 19-23

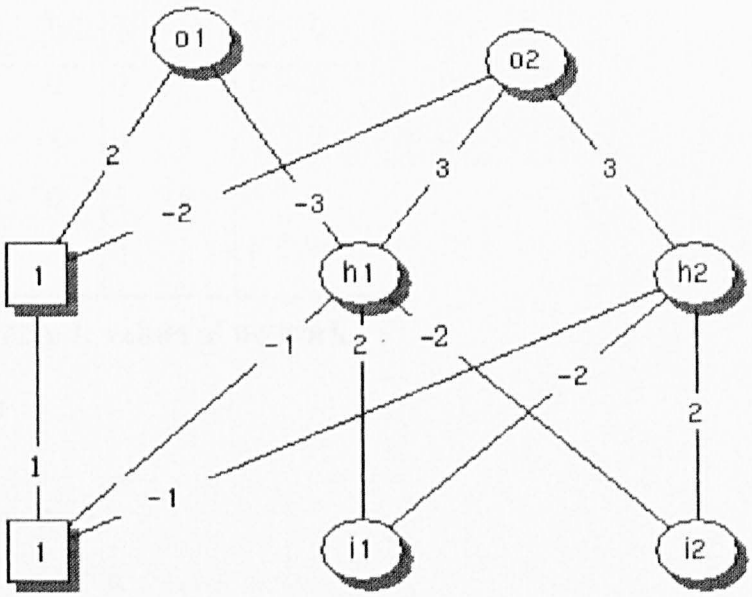


Figure 4: multi-layer network[9]

The output layer therefore is:

$o = f(net2 - \theta_2)$

$net2 = W_2 * h$; (o = output vector, W_2 = matrix of weights of the second level)

Referring to the definition of a single-layer neural net the network in Figure 4 can be expressed as:

$h1 = f(2*i_1 - 2*i_2 - 1)$

$h2 = f(-2*i_1 + 2*i_2 - 1)$

$o1 = f(-3*h_1 + 0*h_2 + 2)$

$o2 = f(3*h_1 + 3*h_2 - 2)$, (f = transfer function).

or with vectors

$$\begin{pmatrix} 1 \\ h_1 \\ h_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 2 & -2 \\ -1 & -2 & 2 \end{pmatrix} \times \begin{pmatrix} 1 \\ i_1 \\ i_2 \end{pmatrix}; \quad \begin{pmatrix} o_1 \\ o_2 \end{pmatrix} = \begin{pmatrix} 2 & -3 & 0 \\ -2 & 3 & 3 \end{pmatrix} \times \begin{pmatrix} 1 \\ h_1 \\ h_2 \end{pmatrix}$$

[9] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 27

i_1	i_2	h_1	h_2	o_1	o_2
0	0	0	0	1	0
0	1	0	1	1	1
1	0	1	0	0	1
1	1	0	0	1	0

Table 1: values of network

or

$h = W_1 * i$

$o = W_2 * h$

In order considerate the threshold we add 1 to the input vector. This bias represents the threshold. Table 1 shows the values of the hidden layer (h_1, h_2) and the output values (o_1, o_2) depending on the input (i_1, i_2). The mathematical rule for a two-layer neural network is

$o= f(W_2 * (f(W_1 *i))$.

Definition (multi-layer neural network)[10]**:** i is the input vector, o is the output vector and h1, h2 ... are vectors of the hidden layers. f is a transfer function and W_1, W_2, W_3 ... are Matrices. A n-layer neural network with input i and output o is

$h_1 = f(W_1*i)$

$h_2 = f(W_2*h_1)$

$h_3 = f(W_3*h_2)$

...

$o = f(W_n*h_{n-1})$

The thresholds are considered in the matrices as bias. The vectors h_1, h_2, h_3 ... represent the hidden layers.

[10] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 29

2.5 The Delta-Rule

To represent a certain function the weights of the neural net must be chosen correctly. For this purpose the real output vector is compared with the desired output vector. Widrow and Hoff developed the following formula: [11]

$$\Delta w_{ij} = \sigma * i_i * \Delta o_j$$

Δo_j is the difference between the desired output and the real output during the learning process. After the weights have been changed with Δw_{ij} another Δo_j is calculated to evaluate another Δw_{ij}. The weights will not be corrected further if the real output equals the desired output ($\Delta o_j = 0$). σ is a constant factor ($1>\sigma>0$) which determines the speed of the learning process. If σ is very small the learning is too slow but if σ is too big the learning algorithm might oscillate between negative and positive values. The following chapters about learning algorithms show how this formula is used to train a neural net[12].

2.6 Feedforward Networks

Generally two types of networks exist; feedback and feedforward networks. Feedback networks use their output as input again while a feedforward network calculates one output value only for each applied input value. The data processing can be seen as a forward flow and therefore this process is called feedforward network. The output vector is not reused as input vector. This type of network is also known as "hetero associative network".[13]

---

[11] Widrow, B. Hoff, M. E.: Adaptive switching circuits; Ire Wescon Convention Record, New York, 1960

[12] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 24

[13] Rieger, Anke: Forschungsbericht Nr. 508/93, Neuronale Netzwerke, 1993, Dortmund, p. 5

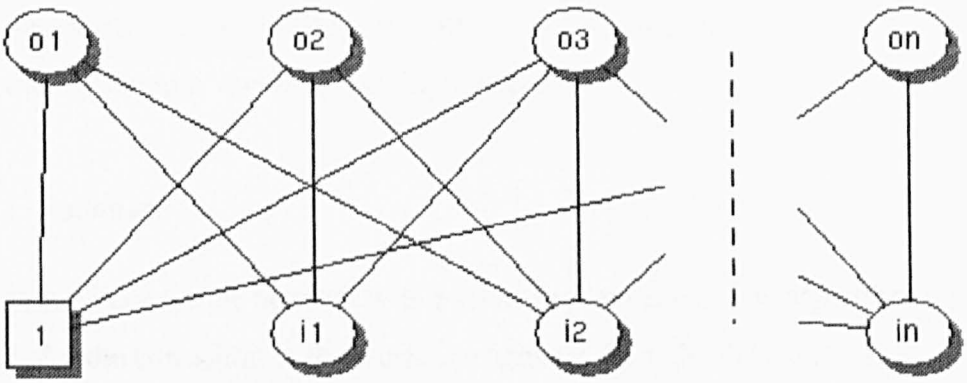


Figure 5: Structure of Adaline[15]

2.6.1 One Layer Network: Adaline

2.6.1.1 Structure

Adaline is an abbreviation for "Adaptive Linear Neuron". The network has got one layer only and the input and output values are binary values of -1 and +1. The general structure is shown in Figure 5.[14]

2.6.1.2 Learning Algorithm

To train Adaline the following supervised learning algorithm is used.

1. Choose random numbers for the weights w_{ij}.
2. Choose any input vector i_j.
3. Change the weights according to the following rule:

   $w_{ij}^{new} = w_{ij}^{old} + \Delta w_{ij}$

   $\Delta w_{ij} = (\alpha * i_j * \varepsilon_i)/(n+1)$, while $\varepsilon_i = o_i - \sum w_{ij} * i_j = o_i - net_i$ represents the difference between desired output and calculated output for i. $1>\alpha>0$ is a small number which corresponds to σ in the Delta Rule (see chapter 2.5).

---

[14] Widrow, B. Hoff, M. E.: Adaptive switching circuits; Ire Wescon Convention Record, New York, 1960

[15] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 32

4. Repeat steps 2 and 3. The algorithm is finished when the network calculates the right output vector for any input vector.[16]

2.6.1.3 Summary

Adaline is a very simple network with a relatively lavish learning algorithm. The formula for the correction of the weights originates from the delta-rule. The learning algorithm increases the weights if net is too little or it decreases the weights if net is net is too big. Not every function can be represented with Adaline (i.e. xor-function).

2.6.2 One-Layer Network: Perceptron

2.6.2.1 Structure

The structure of the Perceptron is similar to Adaline. There are two main differences:

1. The binary input and output values are 0 and 1.
2. To calculate the change of weights Δw during the learning algorithm the output value of the neuron (f(net)) is used instead of the net value (net).

2.6.2.2 Learning Algorithm

Because f(net) is used to calculate the change of weights for the Perceptron there is only the following difference to Adaline in the learning algorithm:

$\Delta w_{ij} = \alpha * i_j * \varepsilon_i$

$\varepsilon_i = o_i - f(\Sigma_k * w_{ij} * i_j) = o_i - \mathbf{f(net_i)}$ [17] (Adaline is calculated with: $o_i - \mathbf{net_i}$)

---

[16] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 33

[17] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 35

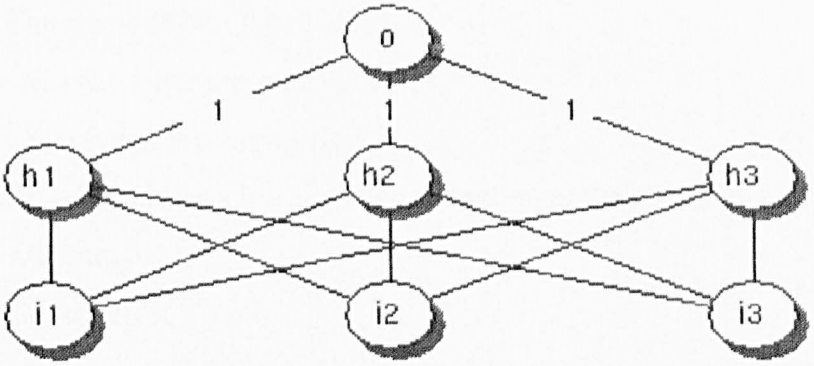


Figure 6: Structure of Madaline

2.6.2.3 Summary

Perceptron and Adaline are very similar and represent very basic and simple networks. Their simplicity helps to understand typical problems of parallel and adaptive systems. As any one-layer network they can be used for linear classification only.[18]

2.6.3 Multi-Layer Network: Madaline

2.6.3.1 Structure

Madaline is an abbreviation for "Multiple Adaptive Linear Neuron". Its structure is similar to the Adaline except there is an output layer with one output element added. The second layer has fixed weights (= 1). Therefore the learning ability of the network is limited to the first layer (Figure 6).

2.6.3.2 Learning Algorithm

1. Choose any input value and calculate the output value o.
2. If o = t (target value), then repeat step one, otherwise go to 3.

[18] Ritter, H., Martinetz, T., Schulten, K.: Neuronale Netze, Addison-Wesley, 1992, p. 32

3. Choose node h of the hidden layer with:
   a) $|net_h|$ has the smallest value
   b) net_h has the wrong sign
4. for this node do a learning step according to the learning algorithm of Adaline.
1. Go to step 1.[19]

2.6.3.3 Summary

With this learning algorithm only that node of the hidden layer is changed where a relatively little change of weight can cause a change of their sign and therefore have a considerable high impact on the output. If this is not sufficient to correct the output value the same learning step can be repeated for another node.

2.6.4 Multi-Layer Perceptron

2.6.4.1 Structure

The multi-layer Perceptron is a multi-layer neural network with different hidden layers. Every element of one layer can be connected with any element of the following layers. Every layer can have a bias. As transfer function usually the squashing function ($f(x) = 1/(1+\exp(-c*x))$) is chosen (Figure 7).

2.6.4.2 Summary

Any logical function can be represented with the network. Because less weights need to be corrected networks with many neurons and a few connections only can be trained faster than networks with a few neurons and many connections.[20] Any

---

[19] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 38

[20] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 39

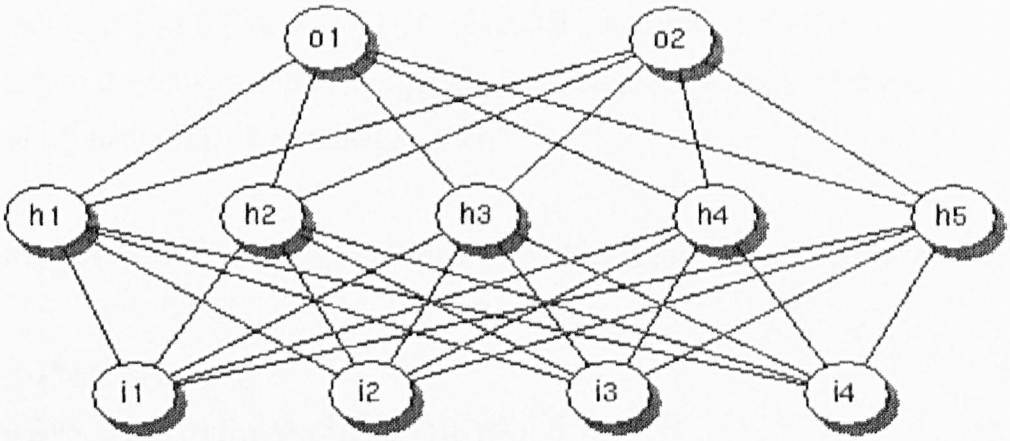


Figure 7: Multi-Layer Perceptron

function that transfers n input values to k output values can be represented with a two-layer neural network.

2.6.4.3 Learning Algorithm: Backpropagation

Because a two-layer network is sufficient to represent any function the Backpropagation-method is explained for a two-layer network. The formulas to calculate the output vector are

(1) $o = f(W_1 * h)$
(2) $h = f(W_2 * i)$.

In order to differentiate we use the squashing function (3) $f(x) = 1/(1 + \exp(-c*x))$ as transfer function. The error of the network is calculated with (4) $E = \frac{1}{2} * \Sigma_i (a_i - o_i)^2$ while a is the aimed value and o the real output value. To improve the performance of the network E should become smaller (if possible 0) during the learning algorithm. If the weights (Δw^1_{ij} for W_1 and Δw^2_{ij} for W_2) are adjusted to the following rule E will decrease:

$\Delta w^1_{ij} = \alpha * \varepsilon_i * h_j$
$\Delta w^2_{ij} = \alpha * \Sigma_m i_m * w^1_{mi} * i_j$

If you put equations (1), (2) and (3) into (4) and if you correct the weights according to the above mentioned algorithm you can prove with the gradient descending method that E becomes smaller.[21]

The formulas for Δw^1_{ij} and Δw^2_{ij} can be improved by the following algorithm:

$$\Delta w^1_{ij} = \alpha * \varepsilon_i * o_i * (1-o_i) * h_j$$

$$\Delta w^2_{ij} = \alpha * \sum_m \varepsilon_m * o_m * (1-o_m) * w^1_{mi} * h_i * (1-h_i) * i_j$$

This correction of the weights is more precise because the weights will be hardly changed if the neurons are close to their aimed values (0 and 1). If the neurons are indifferent (about 0.5) the change of weights is relatively high.

2.6.4.4 Summary Of Backpropagation

The advantage of the Backpropagation method is that there is a mathematical formula which can be used for any network. There is no need to consider any properties of the represented function. Unfortunately the number of learning steps is very high and therefore the learning phase can last very long.

The main disadvantage of Backpropagation is that the error-function is minimised. As in any other mathematical differentiation only a local minimum could be found although the absolute minimum is necessary to represent the right function. If only a local minimum is found many output values are close to the desired output values but they will not match them exactly.

If the number of hidden nodes is too little the desired function possibly cannot be represented because the capacity of the network is not sufficient. If the number of neurons in the hidden layer is enlarged the number of independent variables of the error-function is enlarged, too. This leads to a larger number of minima and the network might run into one of these local minima during the learning algorithm.

---

[21] Hinton, G. E., Rumelhart, D. E., Williams, R. J.: Learning Representations by Backpropagation Errors, Nature 323, 1986, p. 533-536

Furthermore the learning phase lasts longer with each neuron that is added to the hidden layer. [22].

2.6.4.5 Momentum Method

The purpose of the momentum method is to accelerate the convergence of the error back-propagation learning algorithm. The method involves supplementing the current weight adjustments with a fraction of the most recent weight adjustment. This is usually done according to the formula: [23]

$$\Delta w^{new}_{ij} = \mu * w^{old}_{ij} + (1-\mu) * \Delta w_{ij}$$ [24]

$0<\mu<1$ helps to avoid oscillating at the beginning of the learning phase because if w changed it sign during each step then w^{new}_{ij} would be very small because w^{old}_{ij} and Δw_{ij} almost eliminate each other. During the learning process μ increases learning speed because w^{old}_{ij} and Δw_{ij} raises the amount of w^{new}_{ij}.

This is essentially a low-pass filter applied to the search direction. The idea is that if rapid local fluctuations are filtered out, the remaining trend will be toward a more global minimum. This is an extremely effective technique, often speeding convergence by several orders of magnitude; yet, it is not a perfect solution. If too much momentum is used, the algorithm will not be able to follow the twists and turns of the error function. [25]

---

[22] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 40-42

[23] Zurada, Jacek M.: Introduction To Artificial Neural Systems, West Publishing Company, St. Paul, 1992, p. 211

[24] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 43

[25] Masters, Timothy: Practical Neural Network Receipes In C++, Academic Press, San Diego, 1993, p. 101, 102

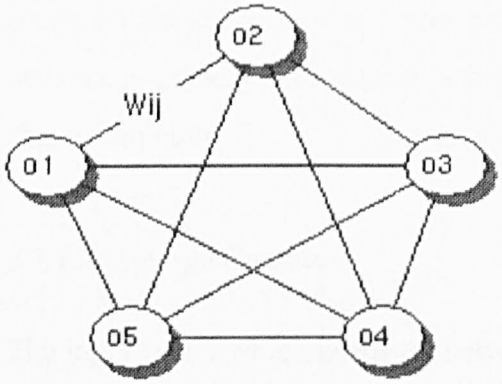


Figure 8: Structure of a Hopfield Network

2.7 Autoassociated Networks

The previous section dealt with networks where one output vector is assigned to one input vector. The data processing can be seen as a forward flow from input to output and therefore this process is called feedforward network. To train the following networks the output is reused as input (feedback). Networks of this structure are called autoassociated networks.[26]

2.7.1 The Hopfield Model

2.7.1.1 Structure

The topology of a Hopfield network is simple. Every neuron is connected with each other (Figure 8). The input values are binary. The following applies for the weights:

1. $w_{ii} = 0$ for every i
2. $w_{ij} = w_{ji}$

The network is calculated with

3. $o_i = f(\Sigma_j w_{ij} * i_j - \delta_i)$

[26] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 50

If o_i is reused as input value for the network new net values are calculated. The network eventually converges which means that output and input values do not change anymore. [27]

2.7.1.2 Energy Function

The input vector which keeps the network in balance (output = input) is called pattern.[28] In a Hopfield network several patterns can be defined. But why does the network recognize a pattern? This question can be answered with the energy function. For each input vector the energy of a network can be calculated. With each iteration the vectors lower their energy potential until a vector is found which represents a local minimum on the energy function[29]. Hopfield networks are mainly used for pattern recognition.

2.7.1.3 Learning Algorithm

To save a pattern it must be described with a n-dimensional vector v. The co-ordinates of vector v are v_j (j = 1, 2, 3, ...n). If you put

$$w_{ik} = \begin{cases} 0 & \text{if } i = k \\ a_i * a_k & \text{if } i \neq k \end{cases}$$

then the network stores the pattern v via matrix $W = (w_{ik})$ and $\delta_k = o$.

The above mentioned rule is used to save one pattern. But usually more than one pattern should be stored in a network. To find out how many patterns can be stored in a network the following rule exists:

A Hopfield network is represented by a matrix W and k patterns and n dimensions. Providing that there is no correlation between the patterns for each

---

[27] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 51

[28] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 53

[29] Hopfield, J.J.:Neural Networks and physical systems with emergent collective computational abilities. Proc. of the Nat. Academy of Science 79, Washington, 1982

input a pattern can be fount with the probability of $p \approx 0,99$ if the number of patterns does not exceed 15% of the number of dimensions ($k \leq 0,15*n$). [30]

2.7.1.4 Summary

Hopfield networks are mainly used for pattern recognition. The prerequisite is that the patterns (e.g. v_1, v_2) do not correlate. The correlation u is calculated with $u = (\Sigma v_1 * v_2)/n$. If u is close to 0 the correlation between v_1 and v_2 is little. In reality patterns unfortunately do correlate. If there is correlation between the patterns there are further minima on the energy function next to the minima which represent the pattern. The network gets stuck into one of these minima which do not represent the function. Getting rid of these minima of the energy function might have a disadvantageous influence on the storing of other patterns. In addition to that Hopfield networks are unable to recognize patterns which are enlarged or reduced or shifted.

2.7.2 Simulated Annealing

2.7.2.1 Procedure

As the section on Hopfield networks described, every minimum of the energy function can represent a pattern. For some optimisation problems the absolute minimum of the energy function is required. Simulated annealing is generally similar to the Hopfield network. During its iteration it runs into a minimum of the energy function. The main difference is in the transfer function. Instead of the binary function the squashing function ($f(x)=1/(1+\exp(-x/T))$) is used. It determines the probability for each vector to reach a lower energy level. While a Hopfield network aims at the nearest minimum during the simulated annealing a vector can "jump" out of the minimum and reach a lower energy level on any place of the function. The squashing function is determined that way that the

---

[30] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 57

probability of leaving a local minimum is very high at the beginning of the iteration. During the process this probability is decreased until the minimum which has been reached then can hardly be left. There is a high probability that the absolute minimum is reached.

2.7.2.2 Summary

Because simulated annealing is used for optimisation problems it is ideal for solving problems such as the travelling-salesman problem.[31] Because simulated annealing is based on a statistical method the absolute minimum is found with a very high probability but not with certainty. In addition to that simulated annealing is used in the field of job-planing (e.g. optimal use of machinery)[32] and packaging problems.[33]

2.7.3 Boltzmann Networks

2.7.3.1 Structure

The Boltzman network is a development of Ackley, Hinton and Sejnowski. Its structure is similar to the Hopfield network[34 35]. However the neurons of the network are divided into three groups (input, output and hidden neurons) (Figure 9).

---

[31] Durbin, R., Willshaw, D.: An analogue approach to the travelling salesman problem using an elastic net method, 1987, Nature 326:689-691

[32] Hopfield, J. J., Tank, D. W.: Kollektives Rechnen mit neuronenähnlichen Schaltkreisen Spektrum der Wissenschaft, Computersysteme, 1989

[33] Dowsland, K. A.: Variants of simulated annealing for practical problem solving, Applications of Modern Heuristic Methods, (ed. V. J. Rayword-Smith) Alfred Waller, Henley-on-Thames, 1995, 3-16

[34] Ackley, D. H., Hinton, G. E., Sejnowski, T. J.L.: A learning algorithm for Boltzmann machines, Cognitive Sciences, 9, 1985

[35] Aarts, E., Korst, J.: Simulates Annealing and Boltzmann Machines, Chichester, 1990

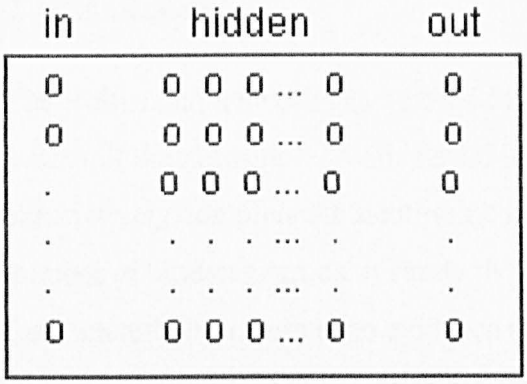


Figure 9: Boltzmann Network

After feeding the network with input values the simulated annealing process is started for the network until the output neurons do not change anymore. During the annealing process the input values must not be changed. The output neurons deliver the result.

2.7.3.2 Learning Algorithm

The basic idea of the learning algorithm is the following:

Simulated annealing is applied

a) only for the hidden cells while input and output values are externally added.
b) for the whole network (the network runs free)

Both procedures are repeated by terms and after each simulation the weights are changed that way that both procedures come closer to each other. The aim is to get those weights that represent the same function during the free run (b) as in (a). If this succeeds the network represents the right function. Because of the very limited use of this type of network no further mathematical algorithms are explained.[36]

[36] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 72

2.7.3.3 Summary

The Boltzmann network can be used for the associated storage of data. This means: If the network fed with partial information only the network can associatively complement the missing information. A problem is to determine the number of hidden neurons. It surely depends on the complexity of the represented function although there is no evidence about the extent. Along with that the learning algorithm requires much calculating.

2.8 Self-organizing Networks

Teupo Kohonen developed a network which method of functioning possibly comes closest to the biological brain.[37]

2.8.1 Sensor Map

2.8.1.1 Structure

If neighbouring spots of the skin are touched the stimulus is sent to neighbouring groups of neurons in the brain. If different stimuli are similar, similar groups of neurons are activated in the brain. There is another example in the field of acoustics. Every frequency activates a certain group of neurons. Neighbouring frequencies make neighbouring groups of neurons active. Kohonen developed a network where always a group of neighbouring neurons reacts on a stimulus. During the learning process the weights of neighbouring neurons are changed with the weight of the mainly activated neuron, too. To implement this feature into a neural network all neurons must be connected with each other that means a feedback network is required.[38]

---

[37] Kohonen, T.: Self Organization and Assoziative Memory, Springer, Berlin, 1988 (2. Edition)

[38] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 83, 84

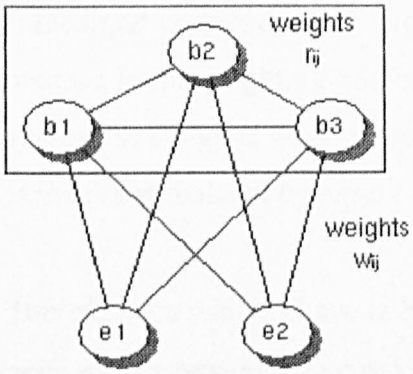


Figure 10: Kohonen Network

In Figure 10 each neuron of the upper layer is connected with the input neurons and each other. Each neuron can be calculated with $b_j = f(\Sigma_k w_{jk}*i_k+\Sigma_k r_{jk}*b_k-\delta_j)$ while r_{jk} is the weight of the feedback.

Feedback coefficients are set up that way that they have a hampering effect over long distances and a stimulating effect for short distances to other neurons in the network. They are positive for neighbouring neurons and negative of distant neurons. If the stimulating or hampering effect is sufficient you can prove that with any input only a neighbouring group of neurons is made active.

According to Figure 10 the following weights can be chosen: $r_{ij} = \exp(-|x_i-x_j|^2/(2*\sigma^2))$. This formula contains the two-dimensional Gauss-distribution with a variance σ^2. x_i and x_j are position vectors and correspond to the neurons in the lattice. The bigger the difference between x_i-x_j the smaller the coefficient. r_{ij} strives for 0 for big distances. Close neurons are strongly influenced. The variance σ^2 determines the spread of the distribution, in this case the radius of influence of the neuron on its neighbouring neurons.

With a biological brain as example the weights w_{ij} should be chosen that way that a net input makes only a local group of neurons active. All other neurons remain uninfluenced.

Furthermore there is the following rule:

If the input vectors are only slightly changed also the location of activated neurons is only slightly changed. Therefore for every input vector i a location vector x is assigned while x represents the location of the neuron in the lattice that is mainly stimulated by input i. Neighbours of x are stimulated, too.

Therefore the weights have to be chosen according to the function $x = \Phi(i)$ which applies for every input vector i the accompanying location vector. If I stands for all possible input vectors i and if A stands for the lattice of neurons then we have the following function: $\Phi: I \rightarrow A$ while $I \subset R^n$ and $A \subset R^2$. Function Φ should be as steady as possible. [39]

2.8.1.2 Learning Algorithm

1. Choose all weights w_{ik}^{in} and w_{ik}^{out} by random.
2. Choose input vector i (stimulus) according to a distribution function p(e).
3. Look for neuron j where $\Sigma_k (i_k - w_{jk})^2$ has the smallest value (centre of stimulation).
4. Improve all weights according to $w_{ik}^{new} = w_{ik}^{old} + \varepsilon * r_{ij} * (i_k - w_{ik}^{old})$ $(\varepsilon>0)$ while j is the neuron from step (3).
5. Go to (2)

For 2) The input values that are used relatively often should also often be used during the learning algorithm. For this data the network is trained better.

For 4) At the beginning the adaptation should be carried out with relatively big values for ε ($\varepsilon \leq 1$). Later on ε is decreased to reach convergence (lim for $\varepsilon \rightarrow \infty = 0$).

Another important parameter for the learning algorithm is the variance σ^2 for the feedback weights r_{ij} [$r_{ij} = \exp(-|x_i - x_j|^2 / (2*\sigma^2))$]. σ^2 represents the range of the feedback. The bigger σ^2 the larger the radius in the lattice of neurons where r_{ij} is

---

[39] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 80

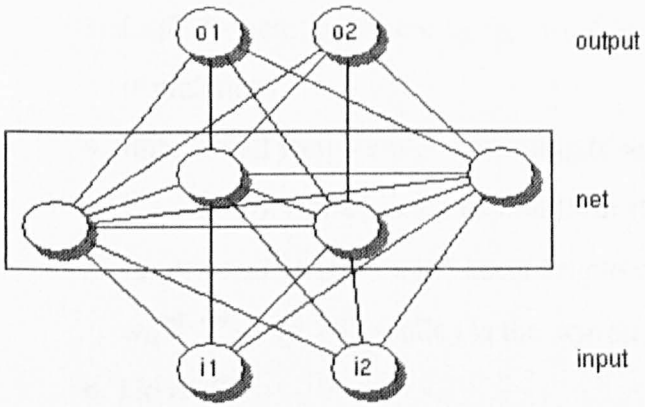


Figure 11: Motor Map

corrected. At the beginning of the learning algorithm a large σ^2 should be chosen to determine the rough structure of weights. After that σ^2 should be decreased to shape a finer structure. The right choice of σ^2 and ε is important for a successful learning. If these parameters become small too quickly the network does not find its global balance. If the parameters converge is too slowly the calculation time is unacceptably long.[40]

2.8.2 Motor Map

2.8.2.1 Structure

The structure of the motor map is very similar to the sensor map. On top of the sensor map is an output layer that uses the activation of groups of neurons on the sensor map as input values (Figure 11). These output values can be used to control an action (e.g. robot control).

2.8.2.2 Learning Algorithm

1. Choose all w_{jk} by random.
2. Choose input vector i (sensor signal) according to a probability distribution p(e).

[40] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 83, 84

3. Look for neuron j where $\Sigma_k (i_k - w_{jk})^2$ has the smallest value (centre of stimulation).
4. Improve all weights w_{ik}^{in} according to $w_{ik}^{in, new} = w_{ik}^{in, old} + \varepsilon * r_{ij} * (i_k - w_{ik}^{in, old})$ ($\varepsilon>0$), while j is the neuron from step (3).
5. Improve all weights w_{ik}^{out} according to $w_{ik}^{out, new} = w_{ik}^{out, old} + \varepsilon' * r'_{ij}{}^{ik} * (o_i - w_{ik}^{out, old})$ ($\varepsilon'>0$), while j is the neuron from step (3).
6. Go to (2)

For 4) The weights between input and network (sensor area) are determined as in the chapter about sensor maps.

For 5) The determination of the output weights equals the determination of the input weights. They are increased if the desired output values are bigger than the actual weights of decreased otherwise. Therefore for the output weights a map exists which has the same structure and characteristics as the sensor map. This map is called " motor map ".

For this learning algorithm input and output values need to be known (supervised learning). If the exact output values are unknown but if an assessment function exists which can judge the quality of the output the network can be described as $w_{ij}^{out, new} = w_{ij}^{out, old} + \varepsilon * r_{ij}$ ($\varepsilon'>0$). r_{ij} is normally distributed around 0 with a variance of 1. Depending on whether the calculated new weights produce a better assessment function they are either kept or rejected. This method is called unsupervised learning.[41]

2.9 Normalisation

Many networks require that all training targets are normalised between 0 and 1 (or -1 and 1) for training. This is because an output node's signal is restricted to a 0 to 1 (-1 to 1) range (transfer functions like the binary function, the signum function, the squashing function, the hyperbolic function, the Sigmoid function, the Gaussian function, the hyperbolic tangent and the hyperbolic secant function can

[41] Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992, p. 94

only have values between 0 and 1 (-1 and 1)). If the output activation function has a range of [0,1], then the target values have to be adjusted in such a way, that they lie within this range. Generally it would be better to choose an output activation function suited to the distribution of the targets than to force the data to conform to the output activation function. For example, if the target values are positive but have no known upper bound, you can use an exponential output activation function. Unfortunately the majority of network simulation programs is made for multiple purposes and therefore it is much simpler to adjust the data to the simulator than to reprogram the network.

Next to normalisation there are other possibilities to transform data. Because most terms are interchangeably used first some definitions:[42]

Rescaling a vector means to add or subtract a constant and then multiply or divide by a constant, as you would do to change the units of measurement of the data, for example, to convert a temperature from Celsius to Fahrenheit.

Normalising a vector most often means dividing by a norm of the vector, for example, to make the Euclidean length of the vector equal to one. In the NN literature, normalising also often refers to rescaling by the minimum and range of the vector, to make all the elements lie between 0 and 1.

Standardising a vector most often means subtracting a measure of location and dividing by a measure of scale. For example, if the vector contains random values with a Gaussian distribution, you might subtract the mean and divide by the standard deviation, thereby obtaining a "standard normal" random variable with mean 0 and standard deviation 1.

Next to the normalisation of output values it can be useful to standardise the inputs, too. For each kind you can distinguish between the standardisation of

---

[42] Sarle, Warren: Frequently asked questions about neural networks, URL: http://www.faqs.org/faqs/ai-faq/neural-nets/part2/, part 2 of 7: Learning, visited on March 16th 1998

column vectors or the standardisation of row vectors. When is which method applicable?

2.9.1 Standardisation of column vectors

If the input variables are combined via a distance function (Kohonen networks), standardising inputs can be crucial. The contribution of an input will depend heavily on its variability relative to other inputs. If one input has a range of 0 to 1, while another input has a range of 0 to 1,000,000, then the contribution of the first input to the distance will be swamped by the second input. So it is essential to rescale the inputs so that their variability reflects their importance, or at least is not in inverse relation to their importance.

If the input variables are combined linearly (Multi-Layer Perceptron), then it is rarely strictly necessary to standardise the inputs, at least in theory. The reason is that any rescaling of an input vector can be effectively undone by changing the corresponding weights and biases, leaving you with the exact same outputs as you had before.[42] However, there are a variety of practical reasons why standardising the inputs can make training faster and reduce the chances of getting stuck in local minima which will be explained in other chapters when necessary.

Very common ways to standardise the columns are[42]:

Notation:
X_i = value of the raw input variable X for the ith training case
S_i = standardised value corresponding to X
N = number of training cases

Standardise X_i to mean 0 and standard deviation 1:

$$mean = \frac{\sum_{i=1}^{N} \mathrm{Xi}}{\mathrm{N}}$$

$$std = \sqrt{\frac{\sum_{i=1}^{N} (\mathrm{Xi} - \mathrm{mean})^2}{\mathrm{N} - 1}}$$

$$\mathrm{Si} = \frac{Xi - mean}{\mathrm{std}}$$

Standardise X_i to midrange 0 and range 2:

$$midrange = \frac{\max Xi + \min Xi}{2}$$

$$range = \max Xi - \min Xi$$

$$Si = \frac{Xi - midrange}{range / 2}$$

Standardisation of **outputs** can be useful if there are two or more target variables and the error function is scale-sensitive , e.g. the least (mean) squares error function, then the variability of each target relative to the others can effect how well the net learns that target. If one target has a range of 0 to 1, while another target has a range of 0 to 1,000,000, the net will expend most of its effort learning the second target to the possible exclusion of the first. So it is essential to rescale the targets so that their variability reflects their importance, or at least is not in inverse relation to their importance. If the targets are of equal importance, they should typically be standardised to the same range or the same standard deviation.
42

2.9.2 Standardisation of row vectors

There are some kinds of networks, such as simple Kohonen nets, where it is necessary to standardise the input cases to a common Euclidean length. Standardisation of cases should be approached with caution because it discards information. If that information is irrelevant, then standardising cases can be quite helpful. If that information is important, then standardising cases can be disastrous. Issues regarding the standardisation of cases must be carefully

evaluated in every application. There are no rules of thumb that apply to all applications. [42]

2.9.3 Non-linear transformation

Most importantly, non-linear transformations of the targets are important with noisy data, via their effect on the error function. Many commonly used error functions are functions solely of the absolute value of the target less the output. Non-linear transformations (unlike linear transformations) change the relative sizes of these differences. With most error functions, the net will expend more effort, so to speak, trying to learn target values for which absolute difference of the value of the target and output is large.

For example, suppose you are trying to predict the price of a stock. If the price of the stock is 10 (in whatever currency unit) and the output of the net is 5 or 15, yielding a difference of 5, that is a huge error. If the price of the stock is 1000 and the output of the net is 995 or 1005, yielding the same difference of 5, that is a tiny error. The net should not treat those two differences as equally important. By taking logarithms, errors are measured in terms of ratios rather than differences, as a difference between two logs corresponds to the ratio of the original values. This has approximately the same effect as looking at percentage differences. That is (abs(target-output)/target) or (abs(target-output)/output), rather than simple differences. [42]

2.10 Conclusion

This section has introduced several types of networks, namely, feedforward networks, feedback networks and self-organising networks which provide the basics for neuroscience. There are also other types of networks which have been developed and implemented and some of these will be illustrated for particular applications in the following chapters.

Generally artificial neural networks are used to compute any computable function; in particular anything that can be represented as a mapping between vector spaces can be approximated to arbitrary precision by feedforward networks.

In practice, neural networks are especially useful for mapping problems which are tolerant of errors, have sufficient example data available, but to which hard and fast rules can not easily be applied.

As so many different types of networks and learning algorithms exist, it can be difficult to decide which type of network and which type of learning algorithm are most suitable for a given task.

This thesis is concerned with cost estimation in project management and the following chapters will give an overview of different types of cost estimating methods and then considers if, and to what extent, neural networks can support cost estimation in project management.

3 Cost estimating in project management

Project management is mainly determined by the following aspects:

1. The required outcome or product of a project;
2. The expected deadlines which are to be met;
3. The resources needed.

At the beginning of a project the monetary input can only be determined by a cost estimate. Unfortunately this cost estimate is often very inaccurate. A very good example for this are the roof of the Olympic Stadion in Munich (20 million Marks estimated costs vs. 171mill. Marks real costs) [43] or the construction of the nuclear power stations in Kalkar ($1.7*10^9$ vs. $6.5*10^9$)[44] or Schmehausen ($0.71*10^9$ vs. $4.4*10^9$)[45].The possible reasons for the error are:

1. Exaggerated optimism by clients and contractors which often results from the toughness of business competition. A more realistic cost estimate can lead to higher trust between promoter and contractor during the contract award process.
2. Changes during the development and production phases which have not been considered during the cost estimate.
3. Insufficient identification of risks[46]

There is clearly a need for more accurate methods of cost estimating.

---

[43] Grün, Oskar: Beiträge zur Projektorganisation, Heft 3: Kosten und Kostenrechnung in der Projektorganisation, Wien, 1977, DBW-Depot 78-3-3 (C.E. Poeschel Verlag Stuttgart)

[44] Der Spiegel no. 14/1982, p. 50

[45] Süddeutsche Zeitung from 4th of June 1982

[46] Madauss, Bernd J.: Handbuch Projektmanagement, Poeschel Verlag Stuttgart, 1990, p. 236

3.1 Cost Analysis

Even though cost analysis and cost estimates look very much alike they represent two fundamentally different processes. During the cost analysis, completed projects are evaluated and analysed. The results represent a basis for future estimates. Methods of estimates are explained later in this chapter. The estimation on the other hand is a prediction of probably arising costs for carrying out a certain new task. The cost analysis is usually very difficult because it is very hard to access usable data material. Furthermore the data material has to be analysed from two points of views:

- from the history of the project point of view
- from the accounting point of view

Furthermore changes of the aim or the planing of the project need to be considered. [47]

Careful analysis of completed projects supply the necessary information for reliable cost estimates of future projects. Cost estimates of new and very complex systems can reach a high quality level if data material is available. In most of the cases these new systems mainly consist of already existing or similar components. A detailed cost analysis is therefore extremely important for the future. [48]

The analysis process should not be restricted on the project organisation level only but also include the hardware, too (e.g. components of the system). Corresponding to the project-structure-plan (see Figure 12) every element of the lowest level needs to be identified and analysed from the cost point of view. The best time for this analysis is right after the conclusion of the project because the relevant information can be easily found and in case of unclarity interviews with members of the project can be hold. During the analysis it is necessary to comment on costs because changes of the project and eventually wrong booking might make an

[47] Madauss, Bernd J.: Handbuch Projektmanagement, Poeschel Verlag Stuttgart, 1990, p. 237

[48] Madauss, Bernd J.: Handbuch Projektmanagement, Poeschel Verlag Stuttgart, 1990, p. 243

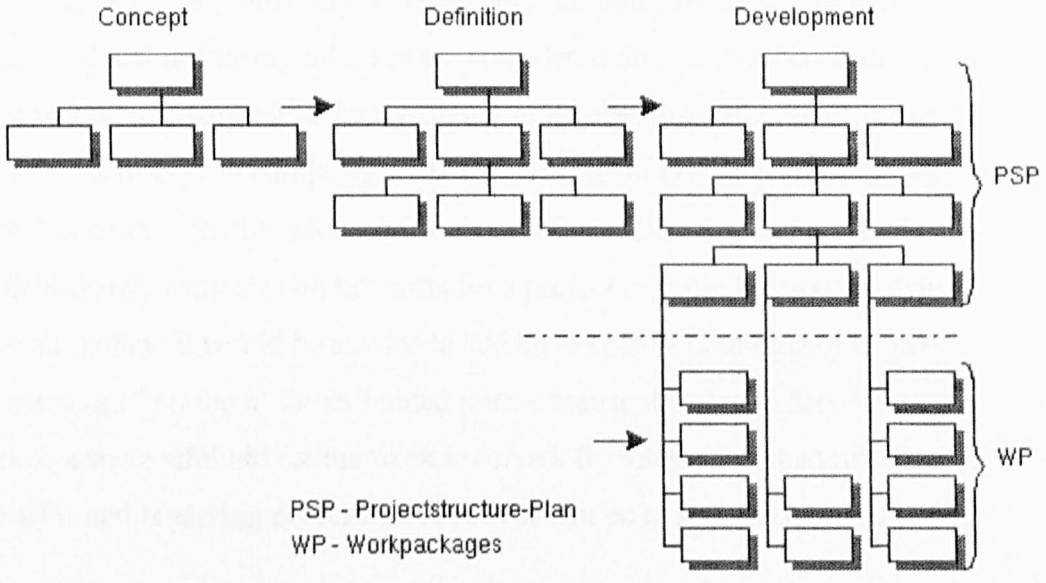


Figure 12: Project-Structure-Plan

objective actual cost determination much more difficult. These explanations should contain:

1. A description of the component with all technical and physical characteristics i.e. the product specification
2. A time schedule (milestones, flow charts and delivery dates)
1. Prototype costs (for developing, testing, production and operation of the first component)
2. Source of information (actual costs, offer price, conditions (fixed price or cost price), contract type)

3.2 Methods of cost estimations

3.2.1 Problems

To provide the most accurate estimate the right information, sufficient time to prepare the estimate and experience from the past is required. These tools make the difference between an estimate and a guess. Naturally the experience of the estimator and the estimating method are very important, too. But even under the best circumstances an estimate remains an estimate. It forecasts the costs of a

project with a certain probability. Even if unforeseeable problems during the development and increasing inflation are considered their real effect on the project can not be predicted exactly. Other, external influences that affect the costs of the project (e.g. strikes) can hardly be predicted. Next to an exact statement about the expected costs the identification of risks is very important. Sometimes the project team deliberately estimates higher costs for a project to avoid the need to request additional money. It would be unwise to add an excessive contingency or even a "fear-surcharge" on top of the estimated price because this would deteriorate the chance to a successful bid for the work involved. By integrating the project team in the sales and tendering process these actions can be to some extent avoided.

There are many forms of contract for project work but only if the price is fixed should a surcharge be considered to cover unpredictable problems. The contract should be set up that way that the project team does not take the responsibility for non-influencable factors (e.g. inflation). Sufficient project control helps to find deviations of estimation at an early stage and helps to start the countermeasures to avoid loss.

Sometimes on the other hand project costs are either deliberately or accidentally underestimated. Both cases usually lead to project management problems, are a burden for every participating organisation and can lead to termination of the project and image loss of involved companies. [49]

3.2.2 Methods

Different methods for the cost estimate of the development and production of a new component exist. Jones and Niebisch summarised them as the following: [50]

1. Judgmental Methods
    - Expert Opinion

---

[49] Madauss, Bernd J.: Handbuch Projektmanagement, Poeschel Verlag Stuttgart, 1990, p. 246

[50] Jones, Ray D. and Niebisch, Klaus: Cost Estimation Techniques, INTERNET Expert Seminar on Cost Control in Project Control, Zürich, 1975

Method / Criteria	Requirements	Application	Disadvantages
1. Judgement - Expert Opinion - Educated Guess - Voting (DELPHI) - Analogies - ROM	- expert experience - rough product definition - analog material	- early phase of project - no risky situation - independent crosscheck - budget estimates - rough planning	- subjective - undefined accuracy - not suitable for detailed negotiations
2. Parametric - CER - Statistical - Probabilistic - Mathematical	- historical data - regression analysis - CER-material	- concept comparisons - budget planing - offer evaluation - independent cross checks - low / medium risk situations	- difficult extrapolation of databases and models - questionable accuracy - extrapolation of database is difficult
3. Detailed Methods - Work Packages - Subcontractor Commitment - Grass-Roots - Computer Approp. - detailed design and material / process data - Bottoms-Up	- time planing - Statement of work - detailed technical material - price quotation	- situation with high risk - price negotiation - CER Calibration - Data Bank Input - Control Estimates	- expensive - time consuming - low flexibility - increases costs (if too detailed) - vulnerable to omissions

Table 2: estimation methods[51]

- Educated Guess
- Rough-Order-Of-Magnitude (ROM)
- Empirical
- Analogy

2. Parametric Methods
   - Cost Estimating Relationships (CER)
   - Statistical
   - Probabilistic
   - Mathematical

3. Detailed Methods
   - Work Packages
   - Industrial Engineering

[51] Madauss, Bernd J.: Handbuch Projektmanagement, Poeschel Verlag Stuttgart, 1990, p. 247

- Grass-Roots

The decision on which method is suitable for the project depends mainly on the phase of the project and therefore on the degree of knowledge about the project. Judgmental methods are suitable before the beginning of a project to get rough cost estimates on the basis of expert opinions. At the beginning of the concept phase, parametric methods should be used. The implementation of detailed methods usually starts in the second half of the definition phase because at this time sufficient planning documents are available e.g. specification, flow charts, work schedules and project-structure-plan.

The tender for the following phases that is set up at the end of the definition phase is usually based on detailed methods. Table 2 shows requirements, applications and disadvantages of each estimation method. Some of them will be explained in the following chapters. Usually the parametric and the detailed method are used in parallel for certain period of time. Because these methods have improved a lot within the last years project managers took the chance to verify their costs by comparing both models. The parametric estimate (top-down-estimate) and the detailed estimate (bottom-up-estimation) therefore complement each other. [51] These and others will be examined in the following section.

3.2.3 Expert Opinion

Each member of a team of experts estimates the costs of a project and presents his results to the team. After exchanging this information every member can change his estimate until a homogeneous result is achieved. Group-dynamic problems might lead to erroneous results, because those members who have a high ability to assert themselves do not necessarily present the best estimate.[52]

[52] Litke, Hans-D.: Projektmangement: Methoden, Techniken, Verhaltensweisen, Hanser Verlag, München-Wien, 1993, p. 119

3.2.4 Educated Guess

Based on its experience, only one person - usually the project manager - roughly estimates the costs of a project. Because this method depends on a single person's knowledge only, it might lead to disastrous results. [52]

3.2.5 Analogy method

The future project is compared with completed projects. This comparison is based on parameters that influence the expenditures. For the comparison those projects should be chosen that meet similar requirements as the future project. Only based on his experience, the estimator has to consider the differences between the projects in the estimate.

Because the analogy method is mainly based on the estimators opinion it is not regarded as objective and it is less easy to justify. People who have not been involved in the original analogy may have problems in interpreting the estimate and in applying it to new projects, where different assumptions about future and past events may be needed.[52]

3.2.6 Relation Method

This procedure is a development of the analogy method. The main difference is that the consideration of differences between projects is based on a formalised procedure. This procedure limits the estimators experience to the selection of comparable projects and their influencing parameters.[53]

3.2.7 Multiplication Method

At the beginning of the estimate, the project is broken down into smaller subprojects. By analysing earlier subprojects the average costs of each subproject

---

[53] Litke, Hans-D.: Projektmangement: Methoden, Techniken, Verhaltensweisen, Hanser Verlag, München-Wien, 1993, p. 120

can be determined. To calculate the total cost the average cost of all subprojects is aggregated.[53]

The disadvantage of this method is, that it does not support the division into partial projects. In addition to that this method assumes a linear connection between the amount of partial projects and the total cost. But empirical data shows an exponential increase of costs in relation to the project size. The reason for this is probably a higher effort into co-ordination and communication.[54]

3.2.8 Rough-Order-Of-Magnitude

The only information that is required for this method are the costs of one phase of the project. To gain this data you either complete one phase and calculate the costs or you undertake a careful estimate. The costs of this phase are compared with an earlier equivalent phase of another project. The proportional deviation of this phase is used as the proportional deviation of the total costs.[55]

For example: The total costs of an earlier project where £10.000.000 and the definition phase of this project cost £1.000.000. If the completed definition phase of a running project costs £2.000.000 the estimated costs of the whole project would be £20.000.000.

Obviously the disadvantage of this method is, that minor mistakes in the estimate of one phase will have a great effect on the total cost estimate. Therefore this method is usually used as plausibility-check for other kinds of estimates.

---

[54] Platz, J.: Projektmanagement erfolgreich einführen, Überlegungen und praktische Erfahrungen beim Aufbau eines Einführungskonzepts, in: zfo, Zeitschrift Führung u. Organisation, 1987, Nr. 4, p.217-226

[55] Litke, Hans-D.: Projektmangement: Methoden, Techniken, Verhaltensweisen, Hanser Verlag, München-Wien, 1993, p. 122

3.2.9 Detailed Cost Estimates Based On Workpackages

Only on a basis of completed planning documents, a detailed cost estimate can provide good results. The Projectstructure-plan (PSP) needs to be broken down to the lowest level and work packages (WP) must be defined (see Figure 12 on page 47). The next step for the preparation of the cost estimate is to describe the work packages which can be seen as mini pieces of work. Combined with a time schedule on this basis the required amounts of material and hours are estimated. WP descriptions inform the reader about the following:

- description of the task
- required information to fulfil this task
- products of the WP

In addition to that a detailed cost estimate requires time schedules, flow charts and technical data (e.g. specifications that define the component). Now an experienced project engineer is able to estimate the required amounts of material based on the available documents with a fairly high accuracy. The project manager multiplies these amounts with adequate costs and summarises them to total costs for each work package.

The amount of material and the costs which result from these for each work package are usually summarised into a cost account. The WP cost accounting is the basis for cost control. For optimal cost control two requirements must be fulfilled. On the one hand they can be calculated from the required amount of material and on the other hand the time distribution of the costs must be obvious. In any case these categories have to correspond with the cost structure of the enterprise. Then for each category the right cost centre of the enterprise can be identified. By multiplying the hourly wage with the estimated hours of work the labour costs can be determined. It is advisable to use gross amounts (hourly wages which include internal surcharges). The advantage of this is that the project manager is able to carry out a gross cost accounting for each work package. Considering a profit margin gives the chance to calculate an bid for the project which is based on work packages. Because each work package is costed and the

time the workpackages are executed can be forecast then it is possible to set up a budget and cash flow plan.[56]

3.2.10 Empirical method

For this method a system of influencing cost drivers is set up and mathematically connected into a formula which is based on empirical data. To calculate the costs of a project the parameters of the cost estimating relationship have to be estimated. Because the formula originates from empirical data, it is almost impossible to locate possible mistakes in the algorithm, and are prone to the difficulties generally encountered by econometric models.[57] Proprietary models have been produced which claim to be applicable to a wide range of project types, and relieve the need for collecting data, fitting curves and so on. These models though need to be calibrated for particular project types, commercial environments and individual organisation.[58]

3.2.11 Parametric Cost Estimates

For parametric cost estimates the right data material plays a significant role. The application of cost estimating relationships assumes a correlation between those component which have been used to establish the CER (Cost Estimating Relationship) and that element which is estimated. The following parameters are frequently used:

- weight	- frequency of a transmitter	- quantity
- speed	- power	- size

[56] Madauss, Bernd J.: Handbuch Projektmanagement, Poeschel Verlag Stuttgart, 1990, p. 249

[57] Litke, Hans-D.: Projektmangement: Methoden, Techniken, Verhaltensweisen, Hanser Verlag, München-Wien, 1993, p. 121

[58] Discussion with R. G. Williams, University of Wales, Swansea, 5th of September 1996

Within the last years parametric cost estimates became more and more important because they represent the only useful model at early phases of the project. [59] During the following phases this method is used as independent tool for verifying detailed methods. The Independent Parametric Cost Estimate (IPCE) - Information is summarised in a special format (DD 2089) which contains the physical and technical data of the component. The US military have used this format since 1982 as a basis for negotiation. If an offer is based on a parametric cost estimate the following criteria has to be considered:

1. The applied CER's have to be structured logically.
2. An easy verification of the implemented CER information must be guaranteed.
3. Between the implemented parameters a sufficient statistical correlation is required.
4. The employed CER's have to be precise and correct.
5. The parametric estimating model or a computer aided estimating systems must be easily verified
6. Before handing in a tender the accordance with the above mentioned criteria should have been checked.

3.2.12 Comparison Of Detailed And Parametric Methods

If an enterprise has decided to use cost estimates systematically, detailed and parametric estimates are the most commonly used methods. Therefore they will be compared with each other in this section. Table 3 shows the characteristics for each method (input, calculation, output, justification). The detailed method requires more precise information about the project as input than the parametric method. Due to this higher amount of information the calculation of costs requires much more effort for the detailed method than for the parametric estimate, which is based on CER's only. The detailed method gives us quite extensive information about project costs, which include labour, material and other cost breakdowns

[59] Madauss, Bernd J.: Handbuch Projektmanagement, Poeschel Verlag Stuttgart, 1990, p. 251

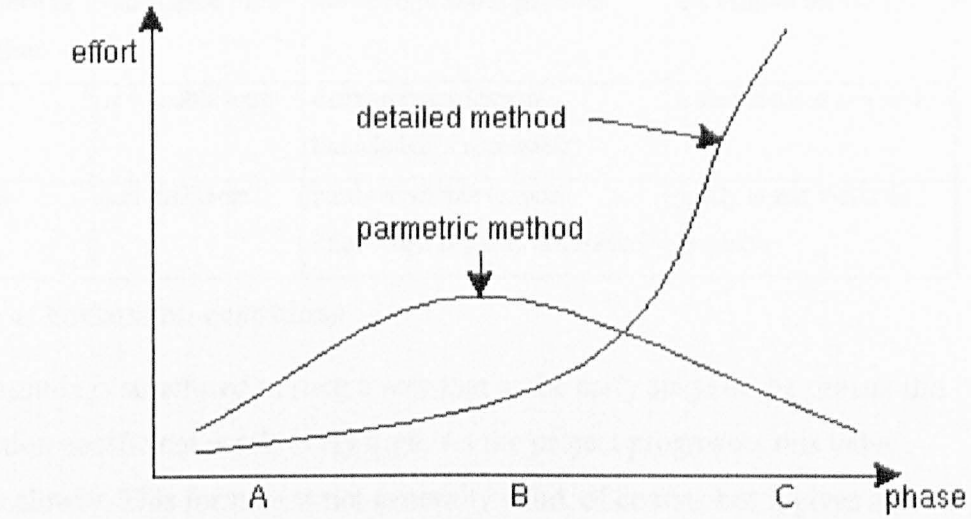
Characteristics	Parametric Method	Detailed Method
Input	Predesign Level, Hardware Definition, Performance Characteristics, Weights and Programmatic Data	Detail Task Definition, Drawings, Specs, Schedules, Materials and Process Date, etc.
Calculation	Cost Estimating Relationships (CER's)	Accumulation of Labour, Material and Other Costs Per Task
Output	Total Cost at WBS Element Level	Detailed Functional Labour, Material, and Other Cost Breakdowns For WBS Elements
Justification	Many Concepts, Limited Level of Definition	Single Concept, Detailed Data Available, Contractual Confidence Required

Table 3: Characteristics of parametric and detailed method

while the parametric estimate is limited to total costs at the WBS (work breakdown structure) element level.

Because the detailed method requires extensive effort it should be used if only a single concept is estimated and a high confidence in the result is required. To compare different project concepts and if the available information is not very precise the parametric model is preferably used. Particularly at the very beginning of the concept phase the enterprise has to choose between different alternatives of the project. Figure 13 shows that at this stage a higher effort is put into the parametric method.

After a few concepts had been chosen to study more carefully, the parametric method is usually still preferred because there is not sufficient information available for a detailed estimate. On the approach to the final concept more and more information is available about this project. This is when the detailed estimate takes over. This method is used to obtain further cost breakdowns in order to set up a cash flow plan or to gather more information for negotiations.



	A	B	C
Status	Many Concepts Low Definition	Few Concepts Medium Definition	Point Design Detail Definition
Method	Parametric Simple CER's	Refined Parametric Detail Backup	Detailed Parametric Check
Objective	Evaluate Concepts Select Preferred	Evaluate Preferred Select Recommended	Refine, Options & Price Recommended

Figure 13: Effort into estimate during concept phase

3.3 Estimating Accuracy

Next to a cost estimate it is important to determine the accuracy of the results. This is important for the consideration of reserves during negotiations. Cost estimates at the beginning are much less precise than those estimates that are made after the definition phase. The reason for this are many unknown influences that are difficult to guess at the beginning of a project. By analysing a large number of projects Augustine[60] developed a correction-factor which is multiplied with the cost estimate to consider these unknown influences.

$$correction\ coefficient = \left(1 + \frac{0.8}{1 + 8t^3}\right)$$

t = r. & d. progress (0 = beginning, 1 = ending of r. & d.)

[60] Augustine, Normann R.: Augustine's Laws and Major System Development Programs, in: Astronautics & Aeronautics, April 1980, p. 36

confidence in estimation	estimation time	knowledge about product	estimation method
high	reasonable long	extraordinary (expert knowledge is accessible)	tested method available
medium	just sufficient	partly available (expert knowledge is partly accessible)	partly tested methods available

Table 4: Estimation conditions

The formula is structured in such a way that at the early stage of the project the correction coefficient is relatively high. As the project progresses this value lowers slowly. This formula is not generally valid, of course, but it gives a good example how unknown influences can be considered. There are further influencing factors that determine the accuracy of a prediction:

1. Circumstances of estimate
    - estimating time (available time for the estimate)
    - estimating team (experienced, partly experienced, inexperienced)
    - prerequisites (economical conditions: hourly wages, expected price fluctuation, profit margin, etc.)
    - assumptions (about the project, operation, etc.)
2. Kind of product
    - technological state of art (established product line, extensive modification, complete design of new technology
    - production experience (high, medium, low)
3. Product description
    - specification status (available, partly available, only rough data available)
    - drawing status (available, partly available, only rough draft available)
4. Estimating method and data bases
    - methods (detailed or parametric estimate)
    - data bases for comparisons (available, partly available, not available)

These factors build a framework to judge the precision of an estimate. Table 4 gives a practical example for a framework that can be used to judge the accuracy of an estimate. In combination with each criteria the confidence (low, medium,

high) is defined. A mark (L, M, H) for the confidence in the estimate of each component is added according to Table 4. In the next step for the three levels of confidence a real value must be examined from experience. This means that components with a high confidence in estimation could for example have a deviation of ±5% of the estimated value. [61]

Experienced project managers may know the general problem well enough to make a detailed and precise cost estimate within the shortest time. Usually the potential client does not consider sufficient time for submitting an offer. But the quality of the cost offer mainly depends on the knowledge about the product, the experience of the estimator, the available data material about preceded projects and the available time for the estimate. Possibly project manager may hardly have time to consult participating departments of subcontractors. Therefore important details could be easily forgotten or other fact could be underestimated. On the other hand the effort for a better quality of cost estimates is very high even if the available time is extremely limited. It is important to apply a tool that supports a faster and more accurate cost estimate.

3.4 Learning Curves

To reach as high accuracy as possible the effect of learning is considered in the cost estimate. Learning curves are used to calculate the average unit cost of a production run. The first unit of a production line is the most costly to produce, with each following unit decreasing in cost at a rate determined by the processing method employed. The cost of the first unit, defined as the first piece cost (FPC) also depends on the production methods used.[62]

Studies indicate that the following are significant factors contributing to the cost/quantity, „learning curve“ phenomena:

- The human learning process;

---

[61] Madauss, Bernd J.: Handbuch Projektmanagement, Poeschel Verlag Stuttgart, 1990, p. 253

[62] PRICE H Reference Manual, PRICE Systems, Moorestown, New Jersey, 1988, 21-1

- Performance experience;
- Production methods, Processes and tooling;
- Quantity;
- Weight;
- Technological Implication[62]

The **human learning process** is the most common given reason for the "improvement" phenomenon. It is generally accepted that the more one produces, the more efficient one becomes. This implies that where human effort or intelligence is involved, there is a learning process, and that repetitive actions will naturally lead to greater efficiency. The production learning improvement is virtually non-existent for automated manufacturing methods which require no human contributions or manual operations. Between total manual fabrication and total automation are varying levels of automation, each affecting the average production cost. [62]

Performance experience relates to the establishment of production standards. In addition to actual "learning", comparable histories will influence the manufacturing rate. Most production standards are established early in the program and once set are seldom changed. It is not uncommon to find standards that are union or peer group regulated. Whether formal or informal, production standards control manufacturing costs. In most cases production standards can only be changed through major revisions to processes and/or tooling. [62]

Production process and tooling for a product are probably the most significant reasons why production costs differ. Highly mechanised processes are employed to manufacture large production quantities since the total production costs are usually less than what they would be if purely manual methods are selected.[63]

The following examples show how learning curve determining factors can be connected:

---

[63] PRICE H Reference Manual, PRICE Systems, Moorestown, New Jersey, 1988, 21-2

Consider an item weighing a few grams with a low manufacturing complexity factor; for example, the common paper clip. The low **technology factor** coupled with the low **weight** suggests a large production **quantity**. One would produce paper clips only in large quantities employing highly **automated procedures**. This combination results in a low average unit cost with a virtually flat cost improvement curve. At the opposite end of the scale, consider an extremely large, high-technology device such as Space Shuttle. Here the quantities are small, suggesting considerable **human involvement** in its production. This causes the cost improvement curve of the Space Shuttle to be steeper than that of the paper clip.[63]

The Boeing Curve is an established method, represented by the following equations: [63]

$$UC = T_1 * Unit_i^{\left(\frac{LogUNITLC}{Log\,2}\right)}$$

$$TC = T_1 * \sum_{i=1}^{QTY} Unit_i^{\left(\frac{LogUNITLC}{Log\,2}\right)}$$

UC = Cost of $Unit_i$

TC = Total Production Costs

T_1 = Theoretical First Piece Cost

$Unit_i$ = Production unit sequence number

UNITLC = A decimal number between 0 and 1 which determines the rate of cost improvement

3.5 Parametric Cost Estimates

Because parametric cost estimates represent the basis for the following chapters in this thesis, they are explained more detailed in this chapter. They can be defined as following:

Definition: A parametric cost estimate represents a method for determining costs by using cost estimating relationships (CER's - see 3.5.1).

3.5.1 Cost Estimation Relationship (CER)

After analysing the development and production costs of a component these costs can be put into relation with certain parameters. Conversely these parameters can help to estimate the costs of this component. In 1969 Batchelder established significant principles for cost estimates that are based on equations. [64] Batchelder says that many estimation equations are formulas which indicate that the costs are proportional to weight, volume or any other physical characteristic of the component. These cost estimating relationships (CER) are especially suitable during the early stages of a project. A detailed cost analysis of already completed projects is the important basis for the derivation of CER's. CER's can only be used if the characteristics of the component that has to be developed or produced are the same, that is they share the same cost drivers.

It is important to consider product groups and other articles with common characteristics. For example if you take the weight of a car as indication of its price you can easily see that the price per kilo of a luxury car is generally much higher than the price of an standard car [65]. The procedure for setting up a CER is the following:

1. Determine simple features which might determine the price of a component.
2. A regression analysis shows the correlation between this characteristic and the production costs.
3. Eventually more characteristics need to be combined to reach a better result.

3.5.2 Project Specific Cost Estimates

These estimates are based on CER's, which can be used for specific applications only. To estimate the costs of a jet engine for example, you might develop CER's

---

[64] Batchelder, C.A./Boren, H.E./Campbell, Jr. H.G./Dei Rossi, J.A. und Large, J.P.: An Introduction to Equipment Cost Estimating, The Rand Corporation, Memorandum RM-6103-SA, December 1969, p. 33, 34

[65] Madauss, Bernd J.: Handbuch Projektmanagement, Poeschel Verlag Stuttgart, 1990, p. 244

which include standing thrust, compression, turbine entering temperature, thrust - weight relation etc. It is obvious that these CER's can be used for jet engines only and therefore it is called a project specific CER. Although this model can be used for specific tasks only its advantage is that the user does not need any historical data because the CER's are already used on historical data of jet engines.

3.5.3 Universal Parametric Cost Estimates

Universal parametric cost estimates exclusively use unspecific - and therefore for many systems usable - parameters. Weight and volume are very common cost drivers as are so-called "complexities" and are used as main parameters. These complexities refer to the difficulty of development or production. Because they are difficult to rate even a little misjudgement can lead to severe errors in the estimate.[66] The level of accuracy depends on the skill of the user.

3.5.4 The Method for Establishing a Parametric Cost Estimating Model

At first all relevant cost elements have to be identified by using the project-structure-plan and cost-structure-plan (see Figure 12 on page 47). According to historical data you can determine to which level of detail the estimating structure can be set up. According to Figure 14 a parametric cost estimating model can be divided into two levels. While the user level does not require any explanation the following sections will look more carefully at the set-up level. Because the following procedures need to be carried out for each CER it can be seen how much labour is necessary to develop a detailed model with hundreds of CER's.[67]

3.5.4.1 Step One: Normalisation of Historical Data

To make the comparison of similar projects or products possible, the database needs to be normalised. Normalisation means to set each data record on the same

---

[66] H. Reschke, H. Schelle, R. Schnopp: Handbuch Projektmanagement Band 1, Verlag TÜV Rheinland Köln, 1989, p. 372

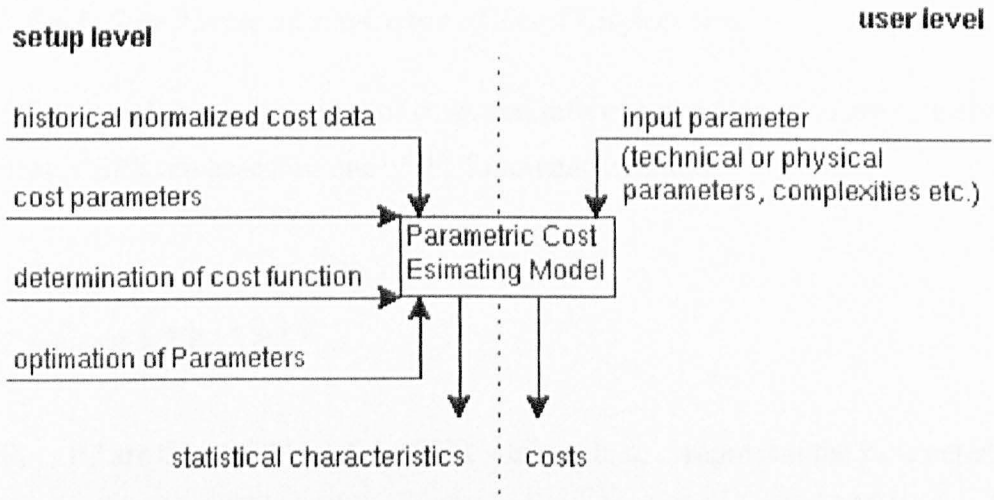


Figure 14: parametric cost estimating model[67]

level to make comparison between the records possible. This includes the consideration of taxes and custom duties. To take account of other currencies not only the exchange rate is important but also its variation, because a high variation of exchange rates might considerable influence the costs. To calculate the costs per unit the effect of the learning curve might be applicable for a huge number of parts. Last but not least the technical progress might devalue the price of older projects.

Because the quality of generated CER's is mainly based on the data material its careful normalisation is of great importance.

3.5.4.2 Step Two: Selection of Cost Parameters

Next to the historical data a corresponding amount of influencing parameters has to be determined. There should be a significant correlation of the parameters and the costs. Therefore these parameters are so called "cost drivers". For example in the civil aviation industry typical cost drivers are weight and performance figures.[68]

---

[67] H. Reschke, H. Schelle, R. Schnopp: Handbuch Projektmanagement Band 1, Verlag TÜV Rheinland Köln, 1989, p. 373

[68] H. Reschke, H. Schelle, R. Schnopp: Handbuch Projektmanagement Band 1, Verlag TÜV Rheinland Köln, 1989, p. 375

3.5.4.3 Step Three: The Selection of The CER function

Even though many functions of costs and influencing parameters are conceivable, most CER's are based on one of the following formulas:

1. $Costs = a + b * P_1^d + c * P_2^e + ...$
2. $Costs = k * P_1^c * P_2^d * ...$

$P_1 ... P_n^d$ are the variables of the CER while a, b, c, ... represent the parameters of the model. The CER's of this must be independent of each other. While the choice of influencing parameters is based on the knowledge of the estimator, the cost estimation relationships (CER) are determined by statistical characteristics.

3.5.4.4 Step Four: Optimisation of Parameters

This term describes the selection of a statistical method to determine the model parameters a, b, c, ... of the CER's from the normalised database of costs. The method of the least squares and multivariate regression analysis are the most common procedures to develop the best fitting curve according to the data material. Next to numbers for a, b, c, ... the result of these methods are also statistical values like the correlation-coefficient R, the standard deviation S and the degree of confidence t which will be described in the following chapter.

3.5.5 Statistical Characteristics

The correlation-coefficient R is a measure for the linear dependency of parameters. The standard deviation S determines the scatter of the data material around the regression curve. The value t describes the degree of confidence of the estimation. All these statistical characteristics are only valuable if a sufficient amount of reference data is available which usually is not the case.[68], [69]

---

[69] Ebner, Clauß: Statistik für Soziologen, Pädagogen, Psychologen und Mediziner, Grundlagen, Verlag Harri Deutsch Thun, 1989

$$R = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}$$

$$S = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2}$$

$$t = \frac{1}{\sqrt{n-3}}$$

3.5.6 Documentation

Next to providing the established CER's it is very important to comment the development of the equation to inform the user about the extent the validity of all implemented parameters.

3.5.7 Accuracy, Random Estimating Errors and Systematic Errors

It is very difficult to predict the accuracy of a parametric cost estimate (see chapter 3.3) because some influencing parameters simply cannot be predicted. On the other hand the statistical value t is a useful parameter to "estimate the accuracy of the estimate" (e.g. " there is a probability of 90% that the costs will be between $40,000 and $80,000 but there is only a probability of 10% that the costs will be between $58,000 and $62,000).

A random error is the deviation of the nominal value that cannot be influenced. The main reason for this is the statistical variance of the data to develop the estimating model. The higher the variance the higher the expected error. Providing that the probability of an over- or underestimate is normally distributed the result of a detailed estimate (aggregation of many small estimates) is more trustworthy than a single estimate. The reason for this is that the estimating errors of each "little" estimate might be compensated by each other. For example in the aeroplane industry a global estimate with fundamental parameters only has an expected error of ± 35%. If the same project is estimated on the basis of many subsystems the error may be reduced down to ± 10%.

These systematic errors usually occur if significant technological progress has taken place which is not considered in the data material where the estimate is based on. The question if the available CER's still meet the necessary requirements can be answered by experts only.[70]

3.6 PRICE-H

PRICE is a set of parametric models produced by Lockheed Martin. Of the models available PRICE-H which is relevant for hardware applications is the model which is relevant for the purposes of this thesis.

3.6.1 The Basic Idea Of PRICE-H

"The basic idea and philosophy of PRICE is its universal implementation and easy use with guaranteed fast access to data!",

Frank R. Freiman, founder and developer of PRICE[71]. In the sixties the company RCA developed a universal estimating model for hardware components which is based on practical experience and CER's. A homogeneous scale was introduced to judge and classify different hardware components. This scale is based on the complexity of the production of a component and has values between 0 and 10 (the higher the value the more complicated it is to produce the hardware component). The advantage of this scale is that the model is not limited to specific project only but all kinds of hardware component can be estimated (see chapter 3.5.3). The complexity is distinguished between two categories: The development complexity and the production complexity (see Table 5 and Table 6).

A distinction of product categories was set up: Ground environment, Mobile, Airborne, Space Flight and Manned Space Flight. Components for ground environment got the lowest code 1 while components for manned space flight

---

[70] H. Reschke, H. Schelle, R. Schnopp: Handbuch Projektmanagement Band 1, Verlag TÜV Rheinland Köln, 1989, p. 378

[71] Madauss, Bernd J.: Handbuch Projektmanagement, Poeschel Verlag Stuttgart, 1990, p. 255

Equipment	Typical Examples	WSCF	1.0 Ground	1.4 Mobile	1.8 Airborne	2.0 Space	2.5 Manned Space
Antennas	Small, Spiral, Horn Flush,	4	4.75	5.64	6.55-7.04	6.92-7.44	6.92-7.44
	Parabolic	8	5.3	5.5	-	-	-
	Scanning Radar 10-40' Wide Phased Arrays (Less Radiators)	6-8	5.9	6.4	7.0	7.2	7.2
Engine & Motors	Automobile - 100 to 400 H.P.	25-35	-	4.30	-	-	-
	Turbo-Jet (Prime Propulsion)	25-35	-	-	6.6-7.9	-	-
	Rocket Motors	14-15	-	-	6.1-6.5	6.4-7.3	7.2-8.2
	Electric Motors	75-100	4.47	5.08	5.3	5.4-6.3	5.4-6.3
Drive Assemblies	Machined Parts, Gears, etc.	7-10	5.11-	5.5	5.8	-	-
	Mechanisms w/Stampings (Hi Prods)	12	5.24 3.33-3.73	-	-	-	-
Microwave	Waveguide, Isolators, Couplers	11-20	5.4-5.6	5.4-5.6	5.5-5.7	5.5-5.9	5.5-5.9
Transmissions	Stripline Circuitry	9	5.7	5.8	5.9	6.0	6.1
Optics	Good (Commercial)	70-90	5.1	5.4	6.3	6.7	7.3
	Excellent (Military)	70-90	5.4	5.8	7.3	7.8	8.0
	Highest (Add 0.1 per 10% Yield)	70-90	5.9	6.8	8.0	8.3	8.5
Ordnance Fuze	Automated Production	14-20	-	4.3-4.65	4.3-4.65	-	-
	Small Production-Min.Tooling	14-20	-	5.11-5.33	5.11-5.33	-	-
Servo	Mech Drive & Coupling Networks	65-75	5.63	5.63-5.7	5.7-6.26	5.7-6.86	5.7-6.86
Tools	Machine Tools	25-30	4.45-4.52	-	-	-	-
Printed CKT Cards	Paper Phenolic	83	4.1-4.3	4.1-4.3	4.1-4.3	4.1-4.3	4.1-4.3
(Boards Only)	GlassExpoxy, Double Sided (Add 0.2 for 3 Layers & 0.05 forAddn'l) Add 0.1 for Plated-Thru Holes	110	5.3	5.3	5.3	5.3	5.3
Cabling	Multiconductor w/MS Connectors	40	4.9	5.0	5.0	5.1	5.2
	Same w/ Hermetically Sealed Connectors	40	5.1	5.2	5.2	5.3	5.3
Battery	Lead Acid	68-125	4.47	4.49	4.61	4.8-5.4	4.9-5.5
	Nickel Cadmium	75	5.39	5.83	6.73	7.63	8.38
Gyro	Inertial Platform Type	79	6.01	6.56	6.8	6.9-9.1	7.0-9.4
Laser Module			7.6	8.5	9.4		

Table 5: mechanical production complexity

have the highest code 2.5. The remaining categories got codes between 1 and 2.5. Now the basic elements for rating different specification levels was set up.

To estimate the costs of a hardware component with PRICE the following data has to be available: Production amount, amount of prototypes, weight, volume, platform (specification level), production complexity, development complexity, starting date and deadline for development and production. Further data can be optionally added or calculated by the program.

In the seventies PRICE-Systems developed another model for the estimation of software development costs (PRICE-S) and PRICE-L to determine life-cycle-costs. [72]

3.6.2 The Functioning of PRICE-H

PRICE structures the work of a project into the following cost categories:

[72] Madauss, Bernd J.: Handbuch Projektmanagement, Poeschel Verlag Stuttgart, 1990, p. 256

Electronic Equipment Description	Types of Components	1.0 Ground	1.4 Mobile	1.8 Airborne	2.0 Space	2.5 Manned Space
ANALOG:	Discretes	6.264	6.857	7.854	9.597	10.063
Receivers, OP Amps,	IS's	6.415	6.950	8.090	9.780	10.098
Audio, Video, RF, Servo	LSI	6.742	7.274.	8.267	9.894	10.184
Drive, etc.	Hybrids	6.919	7.451	8.446	10.076	10.365
	VLSI	7.096	7.629	8.626	10.259	10.547
DIGITAL:	Discretes	6.032	6.742	7.820	9.309	9.710
	IS's	6.182	6.888	7.940	9.420	9.857
Gates, Registers, Buffers,	LSI	6.474	7.174	8.232	9.564	9.963
Counters, etc.	Hybrids	6.648	7.348	8.411	9.743	10.144
	VLSI	6.823	7.523	8.590	9.924	10.325
DISPLAY WITH CRT:	Discretes	5.920	6.681	7.662	8.871	9.664
	IS's	6.094	6.824	7.771	9.008	9.801
TV, Terminals, Radar	LSI	6.410	7.139	8.087	9.184	9.936
Consoles, Test Units, etc.	Hybrids	6.582	7.312	8.262	9.361	10.113
	VLSI	6.755	7.486	8.438	9.537	10.290
DISPLAY NO CRT:	Discretes	5.801	6.535	7.638	8.841	9.527
	IS's	6.000	6.681	7.727	8.985	9.638
L.E.D.'s, Liquid Crystal,	LSI	6.300	7.000	8.019	9.165	9.744
Indicators,	Hybrids	6.473	7.173	8.197	9.344	9.924
Controls, etc.	VLSI	6.648	7.348	8.375	9.524	10.104
TRANSMITTER:	Discretes	6.470	7.218.	8.090	9.692	10.252
	IS's	6.650	7.368	8.245	9.813	10.369
TV, Radar,	LSI	6.801	7.516	8.397	9.930	10.481
Communications, NAV	Hybrids	6.979	7.695	8.579	10.112	10.664
AIDS, Laser, etc.	VLSI	7.158	7.874	8.761	10.296	10.848
POWER SUPPLIES:	Discretes	5.391	5.978	6.941	7.527	8.494
	IS's	5.548	6.289	7.196	7.642	8.602
	LSI	5.678	6.415	7.368	7.971	8.732

Table 6: electronic production complexity

1. Cost for engineering (C_{EN})
    - Cost of technical drawings (C_{TD})
    - Cost of design (C_D)
    - Cost of system engineering(C_{SE})
    - Cost of project management(C_{PM})
    - Cost of documentation(C_{DO})
2. Production
    - Cost of production(C_{PR})
    - Cost of prototype development(C_{PD})
    - Cost of tools and test equipment(C_{TE})

The finding of development costs starts with the determination of the costs of technical drawings as shown in Figure 15. These costs result from the amount of technical drawings and represent the basis for the determination of development costs because all other cost categories are related to them. The total costs of development C_{TC} can be described as following:

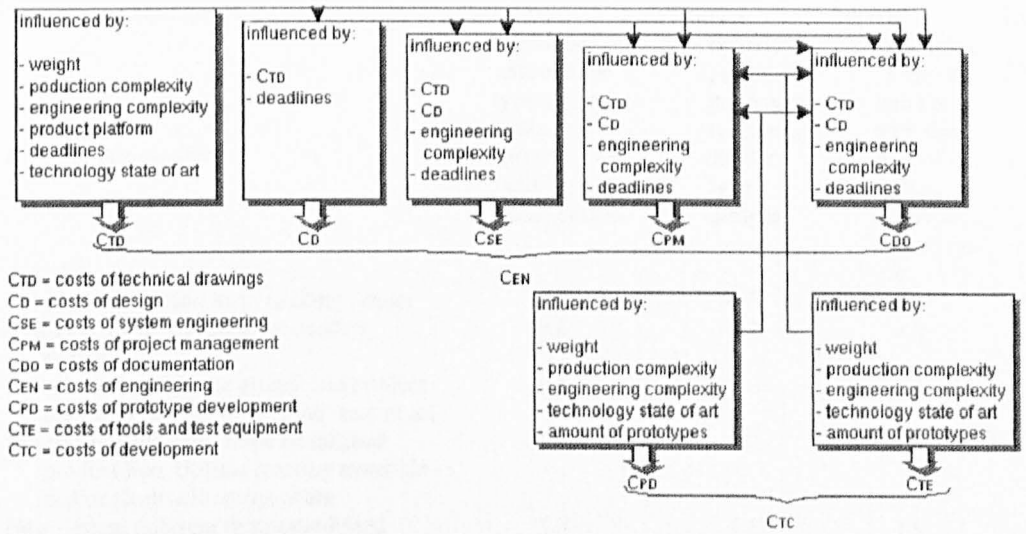


Figure 15: Set-up for calculating development costs

1. $C_{TC} = C_{EN} + C_{MC}$

   where C_{EN} are

2. $C_{EN} = C_{TD} + C_D + C_{SE} + C_{PM} + C_{DO}$

   and C_{MC} is given by

3. $C_{MC} = C_{PD} + C_{TE}$ (manufacturing costs during development; C_{PR} is dropped)

   and C_{TD} is

4. $C_{TD} = SL * EC * TD * AW * CW$

5. SL (specification level): The platform (specification level) describes where the component will be used. The following platforms exist:

- Ground environment (e.g. machine-tool building)	1.0
- Mobile (e.g. vehicle or vessel)	1.4
- Airborne (e.g. plane, helicopter)	1.7-1.8
- Space Flight (e.g. satellites)	2.0
- Manned Space Flight (e.g. Space	2.5

6. The engineering complexity (EC) depends on two parameters: The degree of difficulty of the design and the experience of the personnel. As shown in Table 7 the engineering complexity can vary between 0.2 and 3.1. 0.2 describes a simple modification to an existing design and the personnel is very

Experience of personnel Scope of design effort	Extensive experience, with similar type designs. Many are experts in the field, top talent leading effort.	Normal experience, engineers previously completed similar type designs	Mixed experience, some are familiar with this type of design, others are new to the job	Unfamiliar with design, many new to job
Simple modification to an existing design	0.2	0.3	0.4	0.5
Extensive modification to an existing design	0.6	0.7	0.8	0.9
New design, within the established product line, continuation of existing state of art	0.9	1.0	1.1	1.2
New design, different from established product line. Utilises existing materials and/or electronic components	1.0	1.2	1.4	1.6
New design, different from established product line. Requires in-house development of new electronic components, or of new materials and processes	1.3	1.6	1.9	2.2
Same as above, except state of art being advanced or multiple design path required to reach goals	1.9	2.3	2.7	3.1

Table 7: definition of engineering complexity

experienced. 3.1 depicts a new design that uses new material and production methods while the team is unfamiliar with the job. The following equation shows the ratio of engineering complexity, development schedule and production complexity:

7. $$EC = \frac{DS}{1.3 * PC^{1.1}}$$

DS = development schedule (in months from the beginning of the development until completion of the first prototype)

PC = production complexity (see Table 5)

In Figure 16 the correlation of engineering- and production complexity is shown in relation to the time table and helps the project manager to judge if the time schedule is in a realistic to the fixed engineering complexity.

8. TD (amount of technical drawings): The average amount of technical drawings can be determined by the following equation:

$$TD = \frac{W^{0.7} * PC^{3.7}}{145}$$

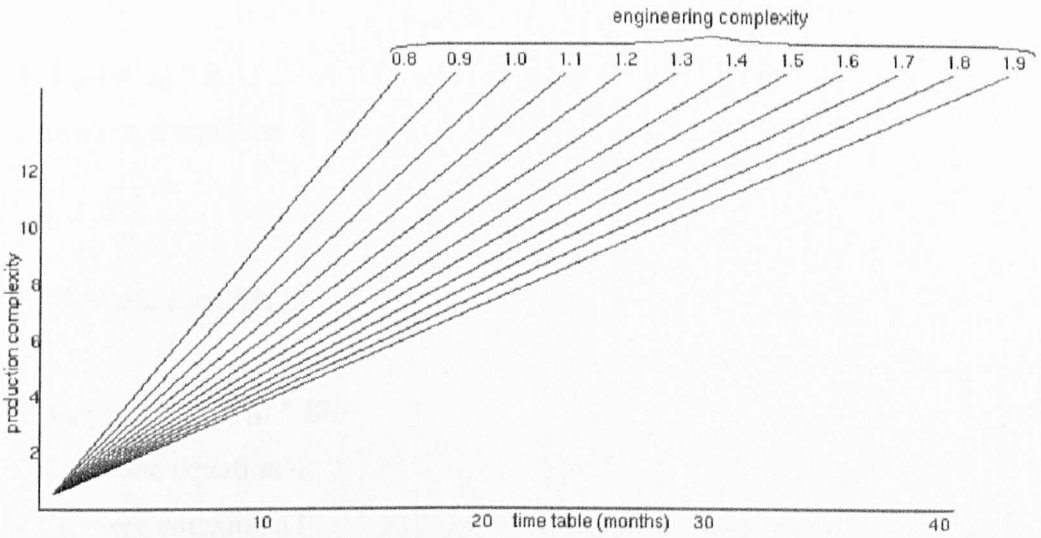


Figure 16: correlation of production- and engineering complexity in relation to the time table

W = system weight (Lbs)

PC = production complexity (see Table 5)

The system weight is either known or can be easily to estimated. On the other hand it is much more difficult to define the manufacturing complexity. To support this determination PRICE developed complexity tables (see Table 5) but still ever enterprise should determine its own values by calibration.

9. AW (average work units per drawing): The amount of average work units per drawing can be derived from the platform. This is obvious that the drawing of a plane requires more work units due to a higher level of checks than any other machine-tool. PRICE suggests the following reference values:

- Ground environment	8 work units
- Mobile	10 work units
- Airborne	15 work units
- Space Flight	25 work units

10. CW (costs per work unit): Because a work unit is defined with 3 men hours the necessary working hours can be easily calculated and multiplied with the hourly wage.

11. $C_D = C_{TD} * R$

C_{TD} = see equation 4

$$R = \frac{DS^{0.3}}{0.7}$$

DS = see equation 7

12. $C_{SE} = (C_{TD} + C_D) * SE_C$

C_{TD} = see equation 4

C_D = see equation 11

$$SE_C = \text{System Engineering Coefficient} = \frac{0.65 * EC^2}{DS^{0.67}}$$

EC = see equation 7 and Table 7

DS = see equation 7

13. $C_{PM} = (C_{SE} + C_{MC}) * PM_C$

C_{SE} = see equation 12

C_{MC} = manufacturing costs during development

$$PM_C = \text{Project Management Coefficient} = \frac{0.4 * EC^2}{DS^{0.67}}$$

EC = see equation 7 and Table 7

DS = see equation 7

14. $C_{DO} = (C_{SE} + C_{MC} + C_{PM}) * DO_C$

C_{SE} = see equation 12

C_{MC} = manufacturing costs during development

C_{PM} = see equation 13

$$DO_C = \text{documentation coefficient} = \frac{0.16 * EC^2}{DS^{0.67}}$$

EC = see equation 7 and Table 7

DS = see equation 7

As shown in Figure 15 on page 70 the costs for technical drawings represent a primary input for the determination of development costs. The developer of

PRICE-H assumed that the amount of technical drawings defines the total development outlay. The expenses for design, development and project management are related in a certain ratio to the costs of technical drawings.

If the development includes the production of a prototype further expenses for the prototype, tools and test equipment apply. Naturally this production also impacts on the project management and project documentation costs as is also shown in Figure 15 on page 70.

In Figure 17 is a flowchart for the determination of production costs. Even during the production phase costs for technical drawings, design and system engineering apply. The reason for this are unavoidable changes that are carried out during the production phase. PRICE-H derives these engineering changes statistically. An overlap of the development phase and the production phase considerably influences the engineering changes because of possible changes of the results of development results. The production costs are mainly influenced by the following factors:

- system weight
- production complexity
- technology state of art
- production quantity
- learning curve
- deadlines[73]

3.7 Conclusion

This section had its main emphasis on parametric cost estimates, as this method is of importance for the following chapters.

[73] Madauss, Bernd J.: Handbuch Projektmanagement, Poeschel Verlag Stuttgart, 1990, p. 258-265

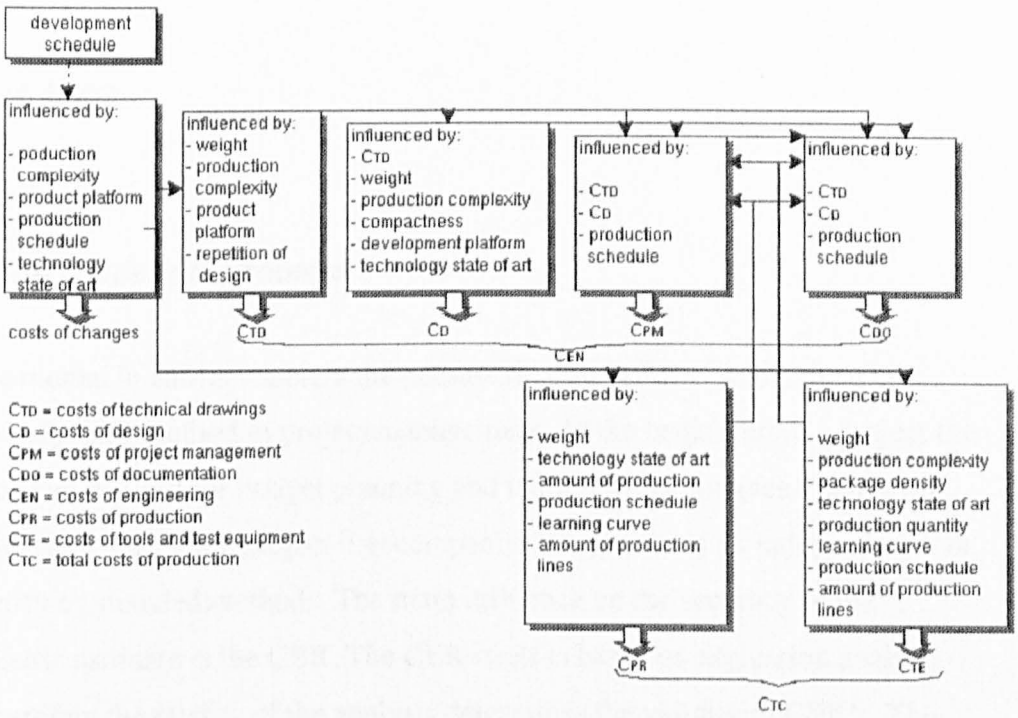


Figure 17: Set-up for calculating production costs

Judgmental methods are the mostly used methods for cost estimating in Europe, because they lead to fast results and they can be cheaply implemented. As already mentioned the major disadvantages are subjectivity and low accuracy.

If a company decides to integrate cost estimation systematically into the organisation structure, parametric cost estimates is the most implemented method. The more it is integrated into the companies organisation the more it represents the basis for negotiations, self control and scenario management.

As well as PRICE there are other commercial providers of computer aided parametric cost estimating models. The author explicitly mentions and concentrates on PRICE because the following sections will focus on the data and structures that are provided by PRICE.

4 The Idea

4.1 The Statistical Problem

As mentioned in earlier chapters the parametric cost estimate is the most commonly used method in project management. At the beginning of a project this method can be used for budget planning and tender evaluation (see Table 2 on page 49) and during the project it accompanies the phases as an independent tool for verifying detailed methods. The main influence on the accuracy of the parametric estimate is the CER. The CER itself is based on regression analysis and therefore the quality of the analysis determines the validity of CER's. This chapter gives a closer insight into possible errors that are based on regression analysis and which affect the quality of the CER and parametric cost estimate. For simplification purpose mathematical formulas and proofs are not included in this section. To provide sufficient information the idea of mathematical procedures is displayed graphically.

4.1.1 Two-Dimensional Example

In real life many cost structures are not linear. For example the driving speed of a car influences the consumption of fuel and therefore the costs. Driving at a very

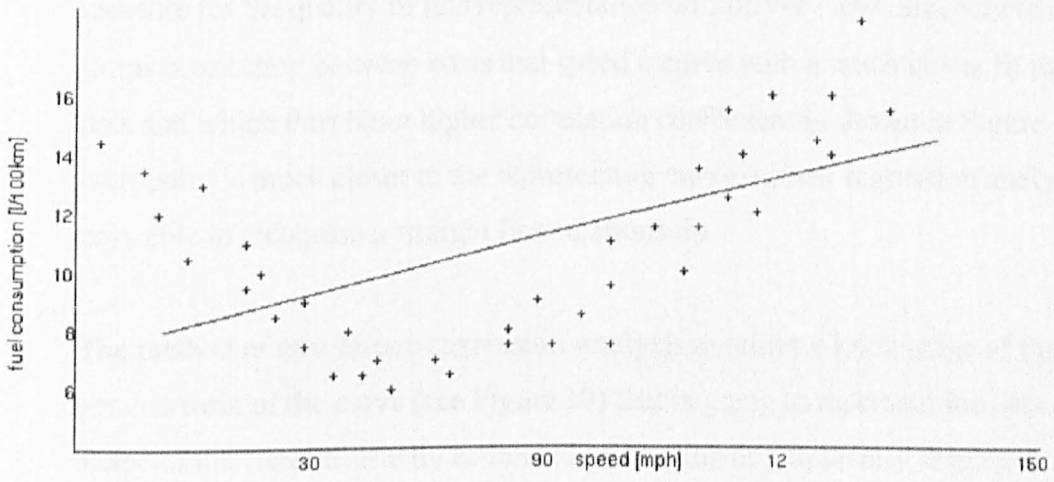


Figure 18: Linear regression analysis of speed and fuel consumption

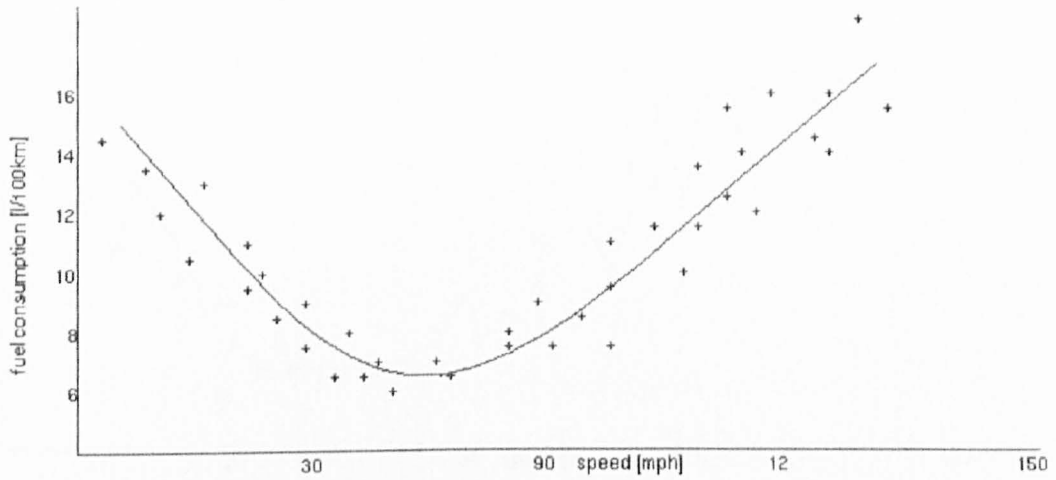


Figure 19: Non linear regression analysis of speed and fuel consumption

low speed usually creates relatively high fuel usage because the car is driven with a low gear. Driving at a higher speed does not necessarily mean a higher usage of fuel because both a higher gear is used and the engine can run at its optimal torque. Driving at a very high speed usually increases the consumption of fuel because wind resistance and engine work rates are relatively high. Each point in Figure 18 is an example for how much fuel has been used at a certain speed.

With the method of the least squares a **linear regression analysis** would supply a straight line to represent the data (see Figure 18). The advantage of this method is that there is a mathematical algorithm which creates the equation of this line and the user of this method does not need to estimate the course of the function. Unfortunately in this case this representation is very weak because many data records are far away from the representing line and the correlation coefficient as measure for the quality of this representation will be very low. Since there is a non linear connection between costs and speed a curve with a much closer fit to the data and which thus has a higher correlation coefficient is shown in Figure 19; each point is much closer to the representing curve. Linear regression analysis is only able to recognise a straight line relationship.

The method of **non linear regression analysis** assumes a knowledge of the general form of the curve (see Figure 19) that is going to represent the data. The shape of the curve is usually estimated by looking at graphically displayed data. If the equation of the curve was familiar there would be no need to carry out a

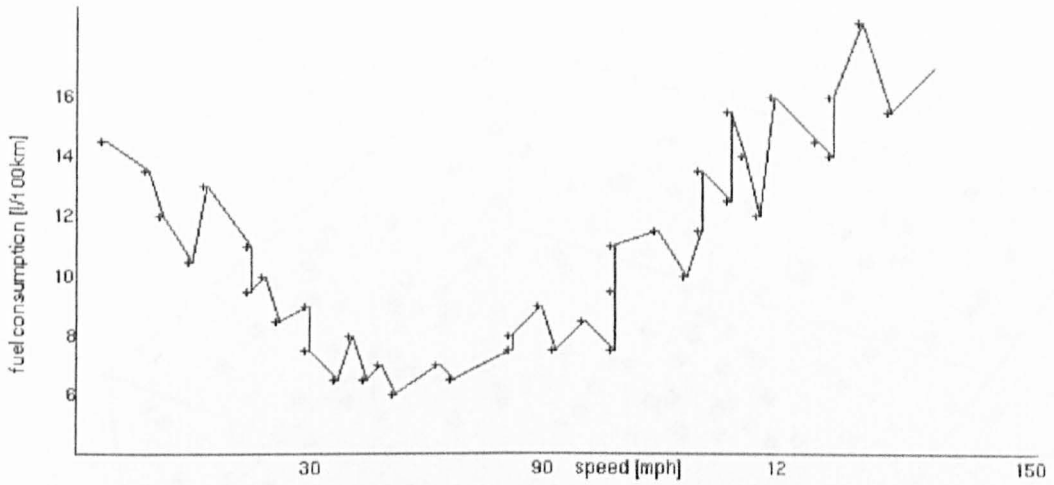


Figure 20: Kernel estimator of speed and fuel consumption

regression analysis! With the method of **co-ordinate transformation** the data points can be transformed into a straight line. With the help of linear regression analysis the best fitting line is found. Although there is an advantage in obtaining a better fitting curve, this method has a major disadvantage. The user has to guess the right transformation according to the data. In the example above this equation can be recognised quite easily because the dots are located very close to each other and a mathematical relationship is fairly clear. Practical experience shows that in many cases it can be very difficult to find the underlying equation.

The **kernel estimator** (or robust estimator) is a statistical method to determine a curve without the necessity of guessing the initial equation. According to Figure 20 the data is divided into many little subsections. For each of these subsections a linear regression analysis is carried out. To retrieve a curve each small line is linked with its neighbouring lines one after another. This method is mathematically extremely complicated and very time consuming, even for specialists in this field[74], and would be impractical for the majority of cost estimators in a business environment.

[74] Discussion with Prof. Dr. Bernd Bluemel, Maerkische Fachhochschule Iserlohn, Hagen, 28th of August 1996

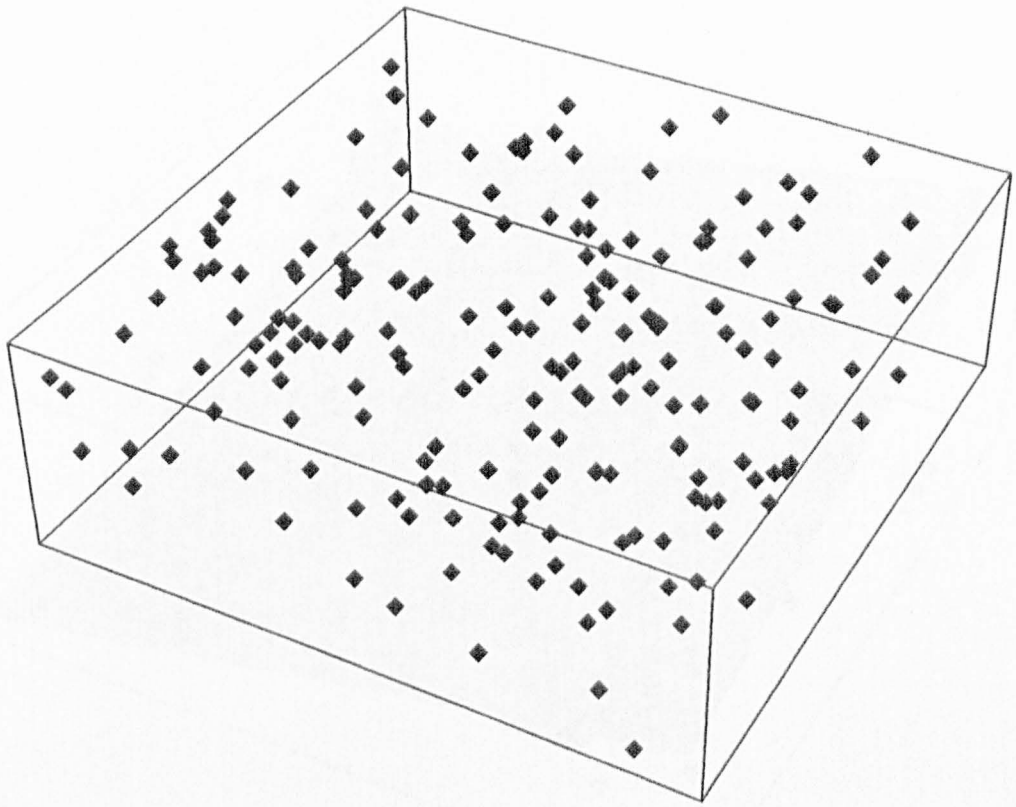


Figure 21: Three-dimensional data sets

4.1.2 Three-Dimensional Example

The example in chapter 4.1.1 was limited to one parameter (speed) that influences the consumption of fuel. Now we look at an example where two parameters affect the costs. Providing that the car drives up or down a hill the fuel usage also depends on how steep the slope is. Driving downhill decreases and driving uphill increases the fuel consumption.

Figure 21 shows how the data might be spread in the three-dimensional space. x- and y-values represent the influencing parameters (speed and slope) while the result is shown on the z-axis (costs of fuel). Comparable to the linear regression analysis in chapter 4.1.1 now the **multiple regression analysis** supplies a plane to represent the data (see Figure 22). Besides the above mentioned advantage (mathematical algorithm to create the equation) again the correlation coefficient can be very poor because many records are far away from the representing plane.

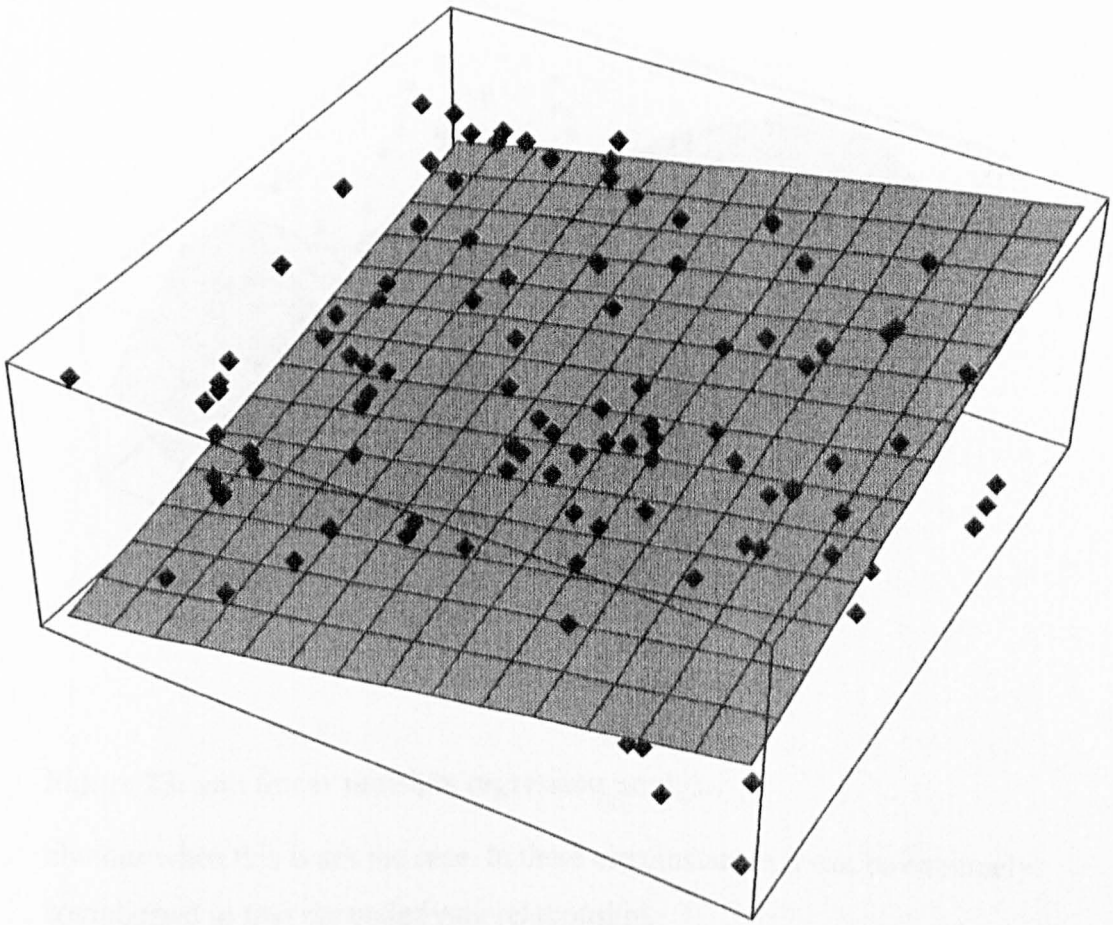


Figure 22: Multiple regression analysis

A mountain range as shown in Figure 23 is much closer to the data group and would therefore reach a better correlation. But unfortunately the result of a multiple regression analysis is always one dimension below the data material. In this example there are three variables (fuel usage, slope, speed) that create a three-dimensional space. The result of the regression analysis is therefore two-dimensional (plane). In the example of chapter 4.1.1 there are only two variables (fuel usage, speed) that create a two-dimensional space and the result of the regression analysis is therefore one-dimensional (straight line).

Equivalent to the non-linear regression analysis in chapter 4.1.1 the **non linear multiple regression analysis** (see Figure 23) assumes the knowledge of the dependencies between the costs and the variables (an equation with two variables that results in costs must be determined). A disadvantage is that the variables are assumed to be independent of each other although in practice it may not always be

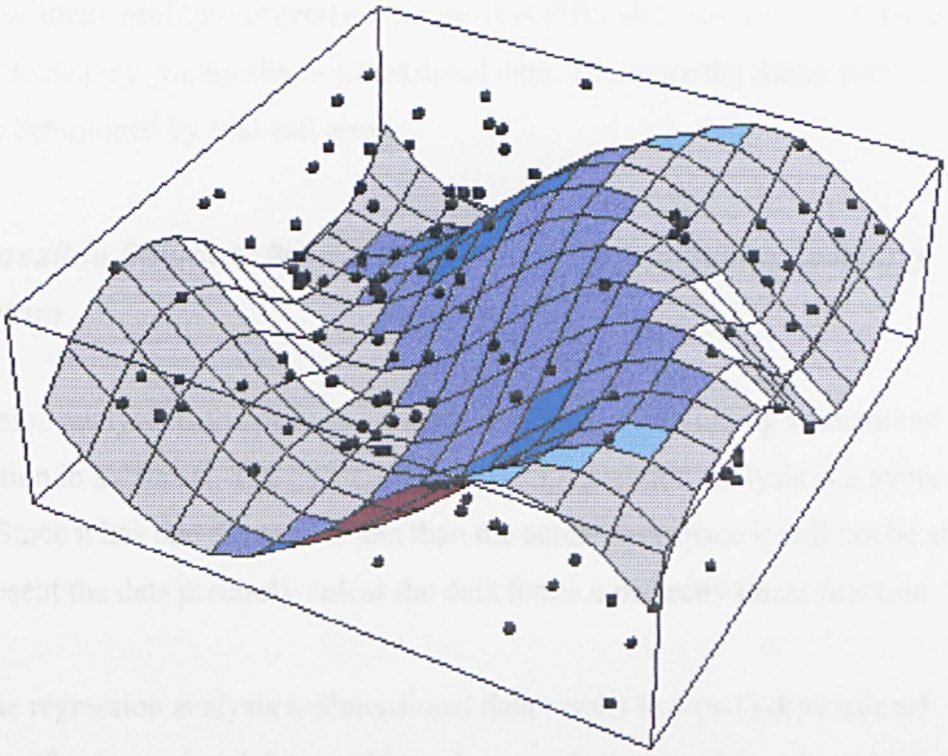


Figure 23: non linear multiple regression analysis

obvious when this is not the case. In these circumstances it can be extremely complicated to find the underlying relationship.

4.1.3 N-Dimensional Example

As well as the speed and slope there are further parameters which influence the fuel consumption of the car - street condition, tyre pressure, weather conditions (wind and temperature) etc.

In this case there are more than three parameters. Their influence on the costs cannot be displayed graphically because it is physically impossible. From the mathematical point of view each extra parameter is treated as an extra variable. Therefore a **multiple regression analysis** can be carried out for n parameters. In chapter 4.1.2 the data was spread in a three-dimensional space. The result of the multiple regression analysis is a plane. In this chapter the data is spread in an n-dimensional space. The result of the regression is a (n-1)-dimensional **hyper plane**. The advantages and disadvantages of the regression analysis still remain.

The **non-linear multiple regression analysis** is difficult to use because there is no way to display graphically n-dimensional data. Therefore the fitting curve can only be determined by trial and error.

4.2 Possible Solution: Neural Net with Backpropagation Learning Algorithm

Regression analysis is the most commonly applied tool for finding a functional connection in a data set. The general product of a regression analysis is a hyper plane. Since it has one dimension less than the actual data space it will not be able to represent the data precisely unless the data forms a perfectly linear function.

With the regression analysis n-dimensional data results in a (n-1)-dimensional function. If n-dimensional data could result in a n-dimensional function a higher correlation can be reached. This is where neural networks begin to play a significant role.

According to chapter 2.6.4.2 a Multi-Layer Perceptron is able to represent any function and there is no limitation to the amount of input and output values. This means that n-dimensional data can be represented with this neural network. Providing that there is a functional connection between the variables a neural network should theoretically be able to find this function. The typical learning algorithm for a Multi-Layer Perceptron is Backpropagation (see chapter 2.6.4.3). Backpropagation is based on a mathematical formula that minimises the error of the network function and there is no need to consider any properties of the represented function. In the following sections the author wants to show to what extent a Multi-Layer Perceptron with Backpropagation learning algorithm is able to replace regression analysis for the determination of a function for parametric estimates.

4.3 Software Programs for Neural Networks

There is wide range of shareware programs for neural networks available. These selected programs offer Backpropagation learning algorithm for Multi-Layer

Perceptron. The following sections summarise descriptions of available software and which were either obtained by the internet[75] or by the help manual of the corresponding software.

1. **Rochester Connectionist Simulator**: A quite versatile simulator program for arbitrary types of neural nets. Comes with a backprop package and a X11/Sunview interface.

2. **GENESIS**: GENESIS 2.0 (GEneral NEural SImulation System) is a general purpose simulation platform which was developed to support the simulation of neural systems ranging from complex models of single neurons to simulations of large networks made up of more abstract neuronal components. Most current GENESIS applications involve realistic simulations of biological neural systems. Although the software can also model more abstract networks, other simulators are more suitable for backpropagation and similar connectionist modelling. Runs on most Unix platforms. Graphical front end XODUS. Parallel version for networks of workstations, symmetric multiprocessors, and MPPs also available.

3. **DartNet**: DartNet is a Macintosh-based backpropagation simulator, developed at Dartmouth, USA, by Jamshed Bharucha and Sean Nolan as a pedagogical tool. It makes use of the Mac's graphical interface, and provides a number of tools for building, editing, training, testing and examining networks.

4. **SNNS 4.1**: "Stuttgart Neural Network Simulator" from the University of Stuttgart, Germany. A luxurious simulator for many types of nets; with X11 interface: Graphical 2D and 3D topology editor/visualizer, training visualisation, multiple pattern set handling etc. Currently supports backpropagation (vanilla, online, with momentum term and flat spot elimination, batch, time delay), counterpropagation, quickprop,

---

[75] Sarle, Warren: Frequently asked questions about neural networks, URL: http://www.faqs.org/faqs/ai-faq/neural-nets/part5/, part 5 and part 6 of 7: free and commercial software, visited on September 9th 1996

backpercolation 1, generalised radial basis functions (RBF), RProp, ART1, ART2, ARTMAP, Cascade Correlation, Recurrent Cascade Correlation, Dynamic LVQ, Backpropagation through time (for recurrent networks), batch backpropagation through time (for recurrent networks), Quickpropagation through time (for recurrent networks), Hopfield networks, Jordan and Elman networks, autoassociative memory, self-organising maps, time-delay networks (TDNN), RBF_DDA, simulated annealing, Monte Carlo, Pruned Cascade-Correlation, Optimal Brain Damage, Optimal Brain Surgeon, Skeletonization, and is user-extendable (user-defined activation functions, output functions, site functions, learning procedures). C code generator snns2c. Works on SunOS, Solaris, IRIX, Ultrix, OSF, AIX, HP/UX, NextStep, and Linux. Distributed kernel can spread one learning run over a workstation cluster.

5. **PDP**: The PDP++ software is a new neural-network simulation system written in C++. It represents the next generation of the PDP software released with the McClelland and Rumelhart "Explorations in Parallel Distributed Processing Handbook", MIT Press, 1987. It is easy enough for novice users, but very powerful and flexible for research use. Works on Unix with X-Windows. Features: Full GUI (InterViews), real-time network viewer, data viewer, extendable object-oriented design, CSS scripting language with source-level debugger, GUI macro recording. Algorithms: Feedforward and several recurrent BP, Boltzmann machine, Hopfield, Mean-field, Interactive activation and competition, continuous stochastic networks.

6. **Uts** (Xerion, the sequel): Uts is a portable artificial neural network simulator written on top of the Tool Control Language (Tcl) and the Tk UI toolkit. As result, the user interface is readily modifiable and it is possible to simultaneously use the graphical user interface and visualisation tools and use scripts written in Tcl. Uts itself implements only the connectionist paradigm of linked units in Tcl and the basic elements of the graphical user interface. To make a ready-to-use package, there exist modules which use Uts to do back-propagation (tkbp) and mixed em gaussian optimisation (tkmxm).

7. **Nevada Backpropagation** (NevProp): NevProp is a free, easy-to-use feedforward backpropagation (multilayer Perceptron) program. It uses an interactive character-based interface, and is distributed as C source code that should compile and run on most platforms. (Precompiled executables are available for Macintosh and DOS.) The original version was Quickprop 1.0 by Scott Fahlman, as translated from Common Lisp by Terry Regier. An early-stopped training based on a held-out subset of data, c index (ROC curve area) calculation, the ability to force gradient descent (per-epoch or per-pattern), and additional options are added. Features (NevProp version 1.16): Unlimited (except by machine memory) number of input Patterns; unlimited number of input, hidden, and output units; arbitrary connections among the various layers' units; clock-time or user-specified random seed for initial random weights; choice of regular gradient descent or Quickprop; choice of per-epoch or per-pattern (stochastic) weight updating; generalisation to a test data set; automatically stopped training based on generalisation; retention of best-generalizing weights and predictions; simple but useful graphic display to show smoothness of generalisation; saving of results to a file while working interactively; saving of weights file and reloading for continued training; prediction only on data sets by applying an existing weights file; in addition to RMS error, the concordance, or c index is displayed. The c index (area under the ROC curve) shows the correctness of the relative ordering of predictions among the cases; i.e., it is a measure of discriminative power of the model.

8. **PYGMALION**: This is a prototype that stems from an ESPRIT project. It implements back-propagation, self organising map, and Hopfield nets.

9. **Basis-of-AI-NN Software**: DOS and UNIX C source code, examples and DOS binaries are available in the following different program sets: backprop, quickprop, delta-bar-delta, recurrent networks, simple clustering, k-nearest neighbour, LVQ1, DSM, Hopfield, Boltzman, interactive activation network, interactive activation network, feedforward counterpropagation, ART I, a simple BAM and the linear pattern classifier.

10. **Matrix Backpropagation**: MBP (Matrix Back Propagation) is a very efficient implementation of the back-propagation algorithm for current-generation workstations. The algorithm includes a per-epoch adaptive technique for gradient descent. All the computations are done through matrix multiplication and make use of highly optimised C code. The goal is to reach almost peak-performances on RISCs with superscalar capabilities and fast caches. On some machines (and with large networks) a 30-40x speed-up can be measured with respect to conventional implementations.

11. **WinNN**: WinNN is a shareware Neural Networks (NN) package for windows 3.1. It incorporates a very user friendly interface with a powerful computational engine. WinNN is intended to be used as a tool for beginners and more advanced neural networks users, it provides an alternative to using more expensive and hard to use packages. WinNN can implement feed forward multi-layered NN and uses a modified fast back-propagation for training. Extensive on line help. Has various neuron functions. Allows on the fly testing of the network performance and generalisation. All training parameters can be easily modified while WinNN is training. Results can be saved on disk or copied to the clipboard. Supports plotting of the outputs and weight distribution.

    Personal comments: This program is easy to use and offers a quite good variety to influence the learning algorithm. In the demo version the amount of nodes is limited.

12. **The Brain**: The Brain is an advanced neural network simulator for PCs that is simple enough to be used by non-technical people, yet sophisticated enough for serious research work. It is based upon the backpropagation learning algorithm.

    Personal comment: Demo version is restricted to number of units the network can handle due to memory constraints on PC's. The software runs under DOS and has no graphical interface.

13. **AINET**: aiNet is a shareware Neural Networks (NN) application for MS-Windows 3.1. It does not require learning, has no limits in parameters (input & output neurons), no limits in sample size. It is not sensitive toward noise in the data. Database can be changed dynamically. It provides a way to estimate the rate of error in your prediction. Missing values are handled automatically. It has graphical spreadsheet-like user interface and on-line help system. It provides also several different charts types.

    Personal comments: Because this program does not require any learning phase, it is probably based on some statistical algorithms which might come close to the properties of a neural network but do not match it exactly. Some tests had been undertaken with this program and the results were quite implausible. Furthermore the learning parameters are limited to one so called "penalty coefficient" and therefore a finer tuning of individual networks is not possible.

14. **PMNEURO 1.0a**: PMNEURO creates neuronal networks (backpropagation); propagation results can be used as new training input for creating new networks and following propagation trials.

15. **NEUNET 1.2**: A Complete Neural Network Development System (personal comments): This program contains the Backpropagation algorithm with only basic parameters to influence the learning. As long as the program is not registered the user is constantly interrupted in his work by a registration reminder.

16. **SLUG**: is a front end for a backprop net with 3 layers, using a sigmoid transfer function in the hidden layer and linear transfer functions in the output layer. SLUG uses the steepest descent optimisation method (simple backprop). You can set the number of nodes in each layer, training parameters and transfer functions. An integrated editor allows instant access to training data and output. SLUG saves networks on dos streams.

    Personal comments: SLUG is a easy to use an offers almost all parameters to

influence the learning process. Unfortunately it has not got any features to show the results graphical display because this helps to understand the results of network more easily.

17. **NNDT 1.4**: The Neural Network Development Tool software is as a tool for neural network training. The MLP algorithm implements multi-layer Perceptron networks. Both feed-forward networks and partially recurrent networks are implemented. Several alternatives for the node activation function are available. The network training is carried out using the Levenberg-Marquardt optimisation method.

    Personal comments: NNDT has a good graphical interface but the software is limited to 3 hidden layers, 20 nodes / layer, and 200 parameters (weights, biases, initial states).

18. **Qnet 2.11**: 32-bit Neural Net modelling under Windows

    Personal comments: Very powerful program with a lot of features but also very expensive ($199)

19. **Fuzzy Symbolic Connectionist Network 1.0**: fSC-Net is a hybrid symbolic/connectionist network that utilises fuzzy logic as its mean to perform uncertainty management. The main purpose of fSC-Net is to act as a knowledge acquisition tool, which can be used by domain experts in the development of expert systems. FSC-Net supports the direct incorporation of fuzzy variable membership functions through the user, or by automatically learning the appropriate membership functions for any given input.

    Personal comments: There is no interactive help menu that supports the use of the program. Therefore it is difficult to use.

20. **TDL 1.1**: The purpose of TDL is to provide users of neural networks with a specific platform to conduct pattern recognition tasks. It is possible to incrementally learn various pattern recognition tasks within a single coherent

neural network structure. Furthermore, TDL supports the use of semi-weighted neural networks, which represent a hybrid cross between standard weighted neural networks and weightless multi-level threshold units. Combining both can result in extremely compact network structures (i.e., reduction in connections and hidden units), and improve predictive accuracy on yet unseen patterns.

Personal comments: The developer of this toolkit is the same as for Fuzzy Symbolic Connectionist Network. Therefore the same problems apply with this program.

21. **Neural Networks for Windows**:

Personal comments: Neural Networks for Windows is a very simple application to understand the basic rules of neural networks. Each neuron and node is graphically displayed. The program is limited to 20 nodes.

22. **NNMODEL 1.06**: The NNMODEL combines a back-error propagation neural network with an advanced statistical based hidden neuron growth heuristic to outperform established methodologies on a wide range of problems while remaining statistically conservative. Statistical and graphical displays enable the user to quickly determine how well the model will perform in its final form. Basic performance statistics, sensitivity analysis reports along with many graphical analysis features round out the NNMODELs analytical tools.

Personal comments: It is an effective modelling tool because it automatically constructs mathematical models directly from the data and it has many features that help to edit and graph the data. In addition to that various statistical tools can help to analyse the data.

This chapter has outlined the use of linear regression in extracting mathematical relationships from raw data. It has also introduced the idea that neural networks can be applied for this purpose. The following chapter develops this further by employing neural networks to such an end.

5 Multi-Layer Perceptron to determine values of the complexity of the PRICE-Model

To determine the manufacturing costs of a component the CER's for PRICE-H are mainly the weight and the complexity (MCPLXS) (see chapter 3.6). The influencing parameters for the complexity are:

- Machinability (describes the difficulty in machining a material)
- Maturity (describes assembly difficulties due to either tight tolerances or expensive labour intensive processes.)
- Platform (establishes the specification and testing level, operating environment, and reliability requirements that the element will be designed to meet.)
- Precision (describes the governing tolerances for the fabricated part or assembly. It should be representative of the labour intensive operations required to obtain the required tolerance.)
- Number of parts (is defined as the number of fabricated parts contained in an assembly, or a reasonable estimate of the number. Fasteners (bolts, nuts, rivets, etc.) should be not included in the parts count.)[76]
- Length (is defined as the longest side of a component. This parameter is put in relation with Precision. For example, a fabricated part with Length = 100mm and a Precision of 1mm is more difficult to produce and therefore has a higher complexity than a component with Length = 10mm and a Precision of 3mm.)

Because machinability, maturity, platform, precision and number of parts are combined in a mathematical function to result in the MCPLXS a Multi-Layer Perceptron should be able to emulate this function. In order to train this neural network Martin Marietta International Inc. Price Systems, Frankfurt, placed the data set in Table 8 at the author's disposal.

[76] Petters, Claus: Computer aided parametric cost estimation, DASA-LA Hamburg, 1995, p. 4-2

Mach	Mat	Plat	Prec	MCPLXS	Mach	Mat	Plat	Prec	MCPLXS	Mach	Mat	Plat	Prec	MCPLXS
100	3	0.6	0.01	3.5168	100	4	1.8	0.07	4.0133	140	3	1.8	0.4	3.6775
100	3	0.6	0.02	3.3248	100	4	1.8	0.09	3.9324	140	3	2.2	0.01	5.6041
100	3	0.6	0.03	3.2173	100	4	1.8	0.2	3.6861	140	3	2.2	0.03	5.127
100	3	0.6	0.04	3.1432	100	4	1.8	0.4	3.4848	140	3	2.2	0.05	4.9192
100	3	0.6	0.05	3.0869	100	4	2.2	0.01	5.3298	140	3	2.2	0.07	4.7869
100	3	0.6	0.06	3.0417	100	4	2.2	0.03	4.876	140	3	2.2	0.09	4.6905
100	3	0.6	0.07	3.0039	100	4	2.2	0.05	4.6784	140	3	2.2	0.2	4.3967
100	3	0.6	0.08	2.9716	100	4	2.2	0.07	4.5526	140	3	2.2	0.4	4.1566
100	3	0.6	0.09	2.9434	100	4	2.2	0.09	4.4608	140	4	1	0.01	3.8615
100	3	0.6	0.1	2.9184	100	4	2.2	0.2	4.1814	140	4	1	0.03	3.5327
100	3	0.6	0.2	2.7591	100	4	2.2	0.4	3.9531	140	4	1	0.05	3.3895
100	3	0.6	0.3	2.6699	100	5	0.6	0.01	3.0948	140	4	1	0.07	3.2984
100	3	0.6	0.4	2.6084	100	5	0.6	0.03	2.8313	140	4	1	0.09	3.2319
100	3	0.6	0.5	2.5617	100	5	0.6	0.05	2.7165	140	4	1	0.2	3.0295
100	3	1	0.01	4.1413	100	5	0.6	0.07	2.6435	140	4	1	0.4	2.8641
100	3	1	0.03	3.7887	100	5	0.6	0.09	2.5902	140	4	1.4	0.01	4.3005
100	3	1	0.05	3.6351	100	5	0.6	0.2	2.428	140	4	1.4	0.03	3.9343
100	3	1	0.07	3.5374	100	5	0.6	0.4	2.2954	140	4	1.4	0.07	3.6734
100	3	1	0.09	3.4661	100	5	1	0.01	3.6443	140	4	1.4	0.09	3.5993
100	3	1	0.2	3.249	100	5	1	0.03	3.334	140	4	1.4	0.2	3.3739
100	3	1	0.4	3.0716	100	5	1	0.05	3.1989	140	4	1.4	0.4	3.1897
100	3	1.4	0.01	4.6121	100	5	1	0.07	3.1129	140	4	1.8	0.01	4.6606
100	3	1.4	0.03	4.2194	100	5	1	0.09	3.0502	140	4	1.8	0.03	4.2638
100	3	1.4	0.05	4.0484	100	5	1	0.2	2.8591	140	4	1.8	0.05	4.091
100	3	1.4	0.07	3.9395	100	5	1	0.4	2.703	140	4	1.8	0.07	3.981
100	3	1.4	0.09	3.8601	100	5	1.4	0.01	4.0586	140	4	1.8	0.09	3.9008
100	3	1.4	0.2	3.6184	100	5	1.4	0.03	3.7131	140	4	1.8	0.2	3.6564
100	3	1.4	0.4	3.4208	100	5	1.4	0.05	3.5626	140	4	1.8	0.4	3.4568
100	3	1.8	0.01	4.9983	100	5	1.4	0.07	3.4668	140	4	2.2	0.01	5.2869
100	3	1.8	0.03	4.5727	100	5	1.4	0.09	3.3969	140	4	2.2	0.03	4.8368
100	3	1.8	0.05	4.3874	100	5	1.4	0.2	3.1842	140	4	2.2	0.05	4.6407
100	3	1.8	0.07	4.2694	100	5	1.4	0.4	3.0103	140	4	2.2	0.07	4.516
100	3	1.8	0.09	4.1834	100	5	1.8	0.01	4.3985	140	4	2.2	0.09	4.425
100	3	1.8	0.2	3.9214	100	5	1.8	0.03	4.024	140	4	2.2	0.2	4.1478
100	3	1.8	0.4	3.7073	100	5	1.8	0.05	3.8609	140	4	2.2	0.4	3.9214
100	3	2.2	0.01	5.6496	100	5	1.8	0.07	3.7571	140	5	1	0.01	3.615
100	3	2.2	0.03	5.1686	100	5	1.8	0.09	3.6814	140	5	1	0.03	3.3072
100	3	2.2	0.05	4.9591	100	5	1.8	0.2	3.4508	140	5	1	0.05	3.1732
100	3	2.2	0.07	4.8257	100	5	1.8	0.4	3.2624	140	5	1	0.07	3.0879
100	3	2.2	0.09	4.7285	100	5	2.2	0.01	5.01	140	5	1	0.09	3.0256
100	3	2.2	0.2	4.4323	100	5	2.2	0.03	4.5834	140	5	1	0.2	2.8361
100	3	2.2	0.4	4.1903	100	5	2.2	0.05	4.3977	140	5	1	0.4	2.6813
100	4	0.6	0.01	3.3058	100	5	2.2	0.07	4.2794	140	5	1.4	0.01	4.026
100	4	0.6	0.03	3.0243	100	5	2.2	0.09	4.1932	140	5	1.4	0.03	3.6832
100	4	0.6	0.05	2.9017	100	5	2.2	0.2	3.9306	140	5	1.4	0.05	3.5339
100	4	0.6	0.07	2.8237	100	5	2.2	0.4	3.716	140	5	1.4	0.07	3.4389
100	4	0.6	0.09	2.7668	140	3	1	0.01	4.108	140	5	1.4	0.09	3.3696
100	4	0.6	0.2	2.5935	140	3	1	0.03	3.7582	140	5	1.4	0.2	3.1586
100	4	0.6	0.4	2.4519	140	3	1	0.05	3.6059	140	5	1.4	0.4	2.9861
100	4	1	0.01	3.8928	140	3	1	0.07	3.5089	140	5	1.8	0.01	4.3631
100	4	1	0.03	3.5614	140	3	1	0.09	3.4382	140	5	1.8	0.03	3.9916
100	4	1	0.05	3.417	140	3	1	0.2	3.2229	140	5	1.8	0.05	3.8298
100	4	1	0.07	3.3251	140	3	1	0.4	3.0469	140	5	1.8	0.07	3.7269
100	4	1	0.09	3.2581	140	3	1.4	0.01	4.575	140	5	1.8	0.09	3.6518
100	4	1	0.2	3.0541	140	3	1.4	0.03	4.1854	140	5	1.8	0.2	3.4231
100	4	1	0.4	2.8873	140	3	1.4	0.05	4.0158	140	5	1.8	0.4	3.2362
100	4	1.4	0.01	4.3353	140	3	1.4	0.07	3.9078	140	5	2.2	0.01	4.9697
100	4	1.4	0.03	3.9662	140	3	1.4	0.09	3.8291	140	5	2.2	0.03	4.5466
100	4	1.4	0.05	3.8055	140	3	1.4	0.2	3.5893	140	5	2.2	0.05	4.3623
100	4	1.4	0.07	3.7031	140	3	1.4	0.4	3.3933	140	5	2.2	0.07	4.245
100	4	1.4	0.09	3.6285	140	3	1.8	0.01	4.9581	140	5	2.2	0.09	4.1595
100	4	1.4	0.2	3.4013	140	3	1.8	0.03	4.5359	140	5	2.2	0.2	3.8989
100	4	1.4	0.4	3.2156	140	3	1.8	0.05	4.3521	140	5	2.2	0.4	3.6861
100	4	1.8	0.01	4.6984	140	3	1.8	0.07	4.2351	140	4	1.4	0.05	3.7749
100	4	1.8	0.03	4.2984	140	3	1.8	0.09	4.1497					
100	4	1.8	0.05	4.1241	140	3	1.8	0.2	3.8898					

Table 8: generated MCPLXS with PRICE

To limit the amount of data the number of parts is set on 1 and has therefore no influence on the MCPLXS. Next to this only two kinds of machinability are used (100 and 140). For each machinability the three most common Maturities (3,4,5) are implemented, which are in a middle of the total range from 1 to 6. All

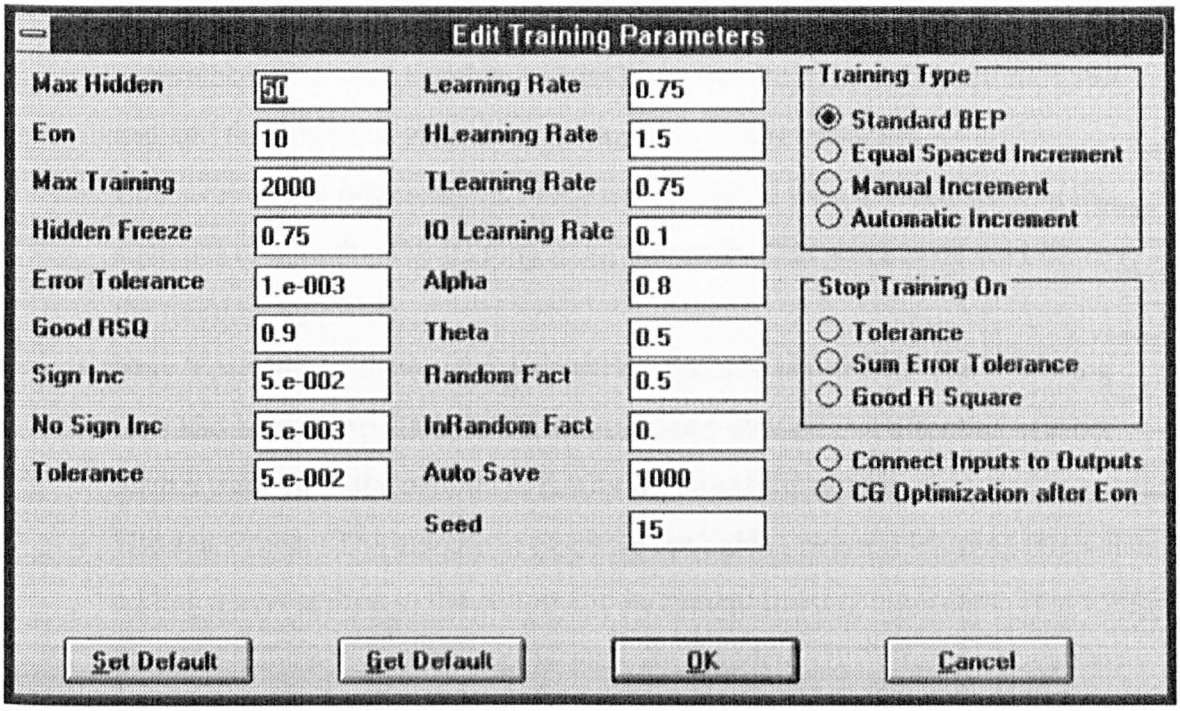


Figure 24: Edit Training Parameters Dialog Box in NNModel

platforms (0.6, 1.0, 1.4, 1.8, 2.2 see chapter 3.6.2) and for each of these platforms the precision is in a range from 0.05 to 0.4 are calculated. Machinability, maturity, platform, precision are input parameters and MCPLXS is the only output parameter.

5.1 The software NNMODEL

According to chapter 2.6.4.2 NNMODEL uses one hidden layer only because this is sufficient to represent any logical function. Since the author of this thesis judges this software program as an effective modelling tool because it automatically constructs mathematical models directly from the data and because it has many features that help to edit and graph the data NNModel will be used to determine values of the complexity of the PRICE-Model.

Before the actual training phase some parameters need to be set up in the "Edit Training Parameters" dialog box (see Figure 24), because some can substantially influence learning behaviour of the network; others have a less significant impact. They are outlined as follows:

- **Max Hidden**: The total number of neurons in the hidden layer is 50. As experience shows this number is quite high. But the training algorithm is still fast and therefore this amount is manageable for the computer.
- **Eon**: Number of presentations of the training set to train before checking the statistics or updating the training progress graph. This value does not influence the training algorithm.
- **Max Training**: Number of steps until the training is finished. The following tests had been set to 2000, because experience showed that after this number further training of the neural net hardly improved the results.
- **Hidden Freeze**: The amount to decrease the hidden neurons learning rate when adding a new neuron in the Automatic Increment training algorithm. This value does not influence the training, because the Standard BEP (Backpropagation) was chosen.
- **Error Tolerance**: The tolerance band that all predictions must be within (total Sq. Error) to end the training. To make sure that this value does not interrupt the training too quickly the tolerance is set to a low value of 0.005.
- **Good RSQ**: When the Rooted Square Error of the model falls under this value, the training is stopped (providing that the "Stop Training On Good R Square"-button is activated).
- **Sign Inc**: Use in calculating when to add a new neuron (Automatic Increment). This value does not affect the training.
- **No Sign Inc**: Used in calculating when to add a new neuron (Automatic Increment). This value does not affect the training.
- **Tolerance**: Acceptable error, used in graphs and in calculating the number of points above and below (training graph) and by the Automatic Increment algorithm. This value does not affect the training.
- **Learning Rate**: The initial learning rate for the hidden layer to output layer connections.
- **HLearning Rate**: The initial learning rate for the input layer to hidden layer connections.
- **TLearning Rate**: The initial learning rate for the all threshold connections.
- **IO Learning Rate**: The learning rate for the direct input layer to output layer connections. Only valid if Connect Inputs to Outputs button is on. The learning

algorithm of the Perceptron ($\Delta w_{ij} = \alpha * i_j * \varepsilon_i$) is based on the Delta-Rule (see chapter 2.5). α is the constant factor which determines the speed of the learning process (if α in the above mentioned formula is very small the learning is too slow but if α is too big the learning algorithm might oscillate between negative and positive values). According to chapter 2.6.1.2 α has a value between 0 and 1.

- **Alpha**: The momentum term (see chapter 2.6.4.5).
- **Theta**: The value feeding all threshold inputs. (default is set to 0.5). This value is not described to the user in the software reference manual. Manual changes of this value do not influence the results of the network and therefore no further attention is paid to this number.
- **Random Fact**: The scaling factor used when initialising the model weights. This value determines the numerical range of the initial weights. Changing this number hardly modified the results of this chapter and therefore nor further attention will be paid to this value.
- **InRandom Fact**: The scaling factor of Gaussian noise used when training the model. This value is useful for the creation of scattering in the data. However in this chapter no Gaussian noise is required.
- **Auto Save**: Number of presentations of the training between auto saving the weights.
- **Seed**: The seed value is used to initialise the random number generator. This value is not further described in the user manual and therefore the author cannot be certain about its specific use. Because different adjustments of this number did not significantly change the performance of the network no further attention will be paid to this value.
- **Training Type:** The following toggles can be selected
  - **Standard BEP**: Train using the standard BEP algorithm.
  - **Equal Spaced Increment**: Add a new hidden neuron at an equally spaced intervals.
  - **Manual Increment**: Add a new neuron to the hidden layer upon command by the user.
  - **Automatic Increment**: Auto detect when to add a neuron to the hidden layer.

- **Stop Training On:** The following toggles can be selected
  - **Tolerance**: Stop training when all predictions are within the tolerance band.
  - **Error Tolerance**: Stop training when the total error is below the value specified.
  - **Good R Square**: Stop when a good enough R square is reached.
- **Connect Inputs to Outputs**: Connect the input neurons directly to the output neurons. This option will speed convergence if the relationships are simple.
- **CG Optimisation after Eon**: Perform conjugate gradient optimisation on the weight matrix after each Eon. CG will not start until at least 500 presentations of the training matrix has been completed.
- **Set Default**: Save the current settings as the default values.
- **Get Default**: Reset the current values to the default.

If the test data are the same as the training data then the model would perform excellently and so a balance needs to be struck between the amount of data used for training and the amount held in reserve for testing the results provided by the neural network.

In order to be able to test the performance of the neural network 10% of the available data shown in Table 8 were chosen randomly and excluded from the training matrix. For the input parameters of this test matrix the output value is calculated by the neural network and compared with the given value of the MCPLXS. This process was repeated four times. The result are five neural models and five test matrices that have been tested with their corresponding model (see Table 9). M_MCPLXS (measured MCPLXS) is the complexity that has been generated by the PRICE-Model. P_MCPLXS is the predicted value by the network and R_MCPLXS (residual MCPLXS) shows the difference between determined and predicted values (R_MCPLXS = M_MCPLXS - P_MCPLXS). If Table 8 is compared with Table 9 you might realise that some of the data sets in Table 9 occur two or three times (shown in bold in Table 9). The reason for this is that the training matrices for each network have been chosen randomly and therefore some data sets have been chosen several times. The predicted values of

M_MCPLXS	P_MCPLXS	R_MCPLXS	M_MCPLXS	P_MCPLXS	R_MCPLXS	M_MCPLXS	P_MCPLXS	R_MCPLXS
2.5902	2.676033	-0.085834	3.615	3.548826	0.066175	**4.1854**	**4.178259**	**0.007141**
2.6084	2.715463	-0.107063	3.6832	3.698549	-0.015349	4.2351	4.293899	-0.058799
2.6813	2.687385	-0.006085	3.7073	3.791609	-0.084309	4.245	4.197404	0.047596
2.7165	2.819942	-0.103441	3.7269	3.80799	-0.081089	4.2694	4.322118	-0.052718
2.7591	2.884526	-0.125426	3.7749	3.721739	0.053161	4.3005	4.151374	0.149126
2.9017	3.025166	-0.123466	3.8055	3.711	0.0945	4.3521	4.460367	-0.108267
2.9861	2.991338	-0.005238	**3.8601**	**3.761904**	**0.098196**	**4.3623**	**4.386294**	**-0.023994**
3.0103	2.98196	0.02834	**3.8601**	**3.752869**	**0.107231**	**4.3623**	**4.366773**	**-0.004473**
3.0243	3.122365	-0.098064	3.8609	3.928773	-0.067873	4.3631	4.371331	-0.008231
3.0417	3.195973	-0.154273	**3.8615**	**3.671534**	**0.189966**	4.3874	4.456241	-0.068841
3.0469	2.98268	0.06422	**3.8615**	**3.675349**	**0.186151**	4.3977	4.380002	0.017698
3.0502	**2.984426**	**0.065774**	3.8989	3.79091	0.107991	4.3985	4.407428	-0.008928
3.0502	**2.951974**	**0.098226**	3.9008	3.926522	-0.025722	**4.4323**	**4.394124**	**0.038177**
3.0716	3.019326	0.052274	**3.9214**	**3.95689**	**-0.03549**	**4.4323**	**4.375638**	**0.056663**
3.0948	3.124415	-0.029615	**3.9214**	**3.933807**	**-0.012407**	4.516	4.467927	0.048073
3.1989	**3.138702**	**0.060198**	3.9306	3.832396	0.098204	4.5466	4.613783	-0.067183
3.1989	**3.118858**	**0.080042**	3.9395	3.852181	0.087319	4.575	4.407325	0.167675
3.2229	3.147342	0.075558	**3.9531**	**3.939078**	**0.014022**	4.6606	4.637693	0.022907
3.2319	3.15843	0.07347	**3.9531**	**3.927917**	**0.025183**	4.7869	4.753033	0.033867
3.2362	3.313623	-0.077423	3.9662	3.874667	0.091533	4.8257	4.764647	0.061053
3.2624	3.339702	-0.077302	3.981	4.014906	-0.033906	4.876	4.916309	-0.040309
3.3072	3.35058	-0.04338	4.0158	3.991331	0.024469	4.9192	4.944203	-0.025003
3.3248	3.558445	-0.233645	**4.0484**	**3.982681**	**0.065719**	**4.9591**	**4.917274**	**0.041826**
3.3739	**3.27237**	**0.10153**	**4.0484**	**3.992234**	**0.056166**	**4.9591**	**4.93781**	**0.021289**
3.3739	**3.284775**	**0.089125**	4.091	4.158689	-0.067689	**4.9697**	**4.896284**	**0.073416**
3.4231	3.464	-0.0409	**4.1413**	**3.954947**	**0.186353**	**4.9697**	**4.857352**	**0.112348**
3.4382	3.366391	0.071809	**4.1413**	**3.94473**	**0.19657**	**4.9697**	**4.884297**	**0.085403**
3.4389	3.424525	0.014374	4.1478	4.0923	0.0555	4.9983	4.978653	0.019647
3.4508	3.482877	-0.032077	4.1497	4.184546	-0.034845	5.6041	5.368817	0.235283
3.4668	3.391402	0.075398	4.1566	4.171335	-0.014735	**5.6496**	**5.455539**	**0.194061**
3.5089	**3.484651**	**0.024249**	4.1814	4.089333	0.092067	**5.6496**	**5.386086**	**0.263515**
3.5089	**3.488857**	**0.020043**	**4.1854**	**4.168245**	**0.017155**			
3.5374	3.479253	0.058147	**4.1854**	**4.166099**	**0.019301**			

Table 9: result of test matrices

these equal data sets differ from each other because their predictions are based on different networks.

5.2 Results

R_MCPLXS in Table 9 shows that there is only a minimal gap between predicted and determined complexities. According to the formulas listed below the relative mean error of the predicted values is only **1.88%** (absolute mean R_MCPLXS is 0.072). The maximum error is 7.03% (R_MCPLXS 0.234). In Figure 25 it can be seen that almost 60% of all errors are in the range from -2% to +2%. An inspection of Figure 25 might indicate that the model tends to fewer large underestimates and more smaller overestimates.

$$\text{absolute mean error} = \frac{\sum_{i=1}^{N} \left| \text{output}_i - \text{target}_i \right|}{N}$$

$$\text{relative mean error} = \frac{\sum_{i=1}^{N} \left| \frac{\text{output}_i - \text{target}_i}{\text{target}} \right|}{N}$$

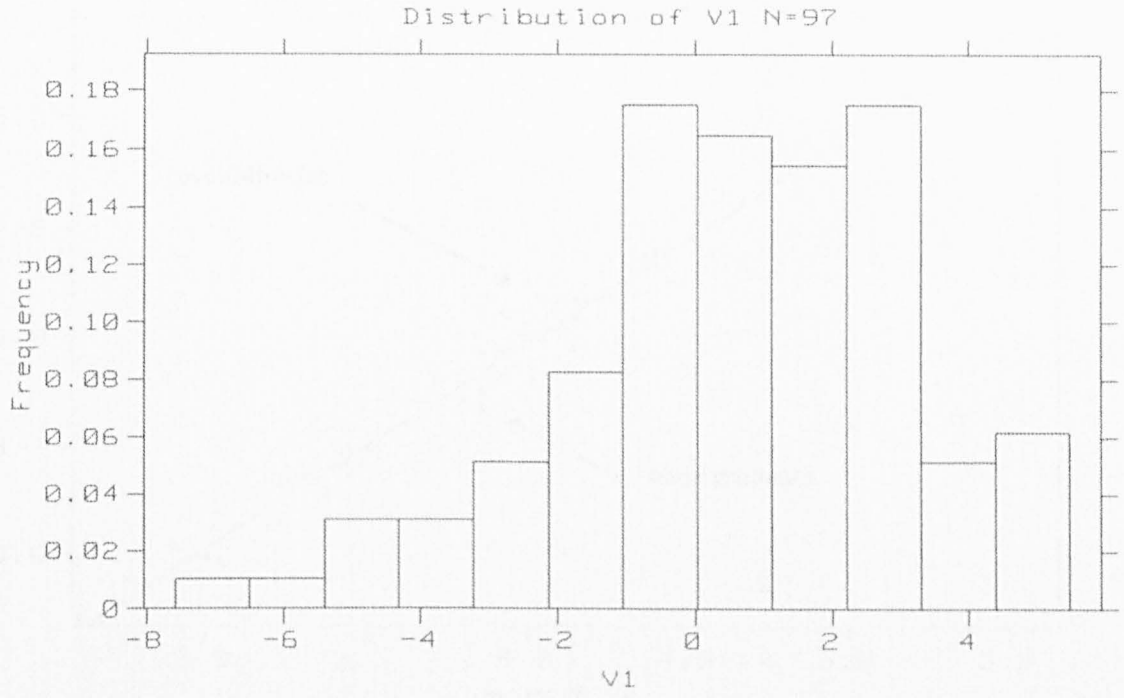


Figure 25: Distribution of relative errors

If you compare all measured complexities with their corresponding predicted complexities in a co-ordinate grid (see Figure 26) the ideal result would be that each point is exactly located on a straight line of gradient 1, i. e. measured value matches the predicted value. Because there are errors in the prediction, most of the dots are not exactly on the line but still very close to it. Here it can be seen that over- and underestimates a quite evenly distributed.

The explanation for the errors, even though they are small, is that the neural model was not able to find the exact equation that represents all vectors precisely. As mentioned in chapter 2.6.4.4 the Backpropagation algorithm might drop into a local minimum that comes very close to the required function but does not match it exactly. This particularly happens if a neural network has many more neurons in the hidden layer than are required. In this example the hidden layer had 50 neurons.

Using the same data on models with 9 neurons in the hidden layer showed that the results of these smaller networks was comparable to the bigger networks. The training time in this case was just less than a tenth of the time that was required

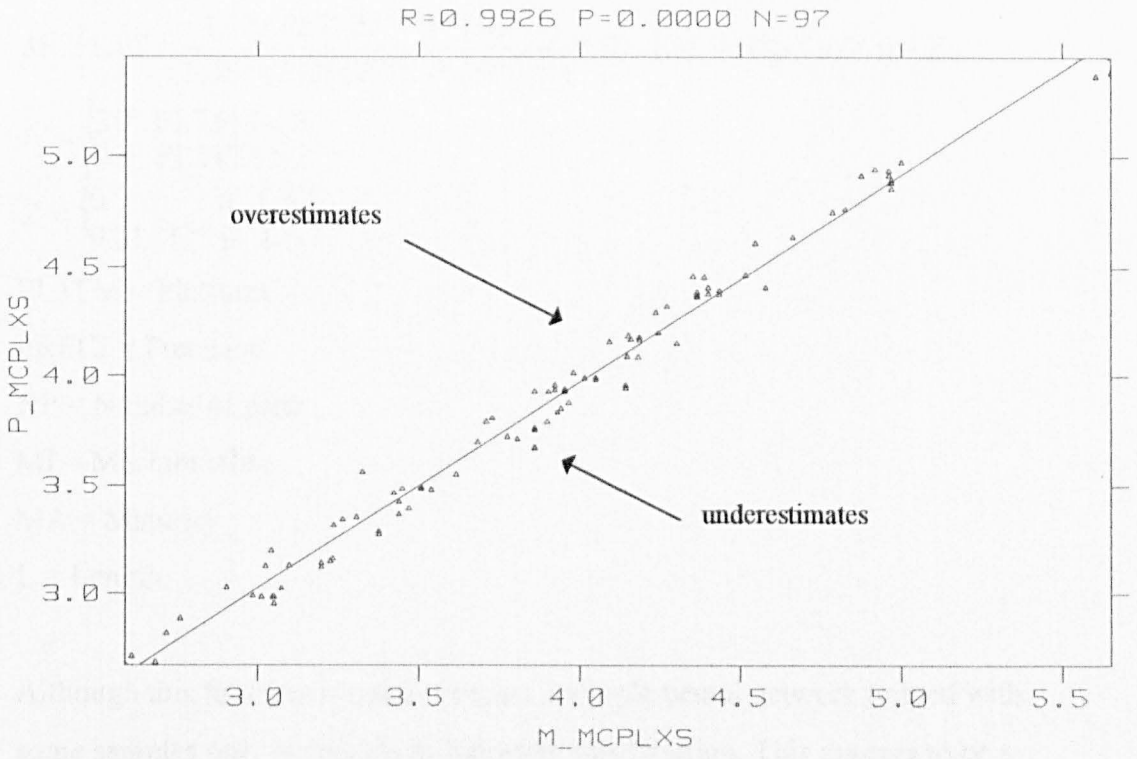


Figure 26: measured versus predicted MCPLXS

for the networks with 50 neurons using the same software and hardware. With a 9 neuron model there was still an average error of about 2%. These networks might have dropped into local minima although this is less likely than for the 50 neuron model.

The good quality of the predicted values is not really very surprising or a reason to be very enthusiastic about neural networks. It must be considered that the complexity is strictly based on a mathematical function with machinability, maturity, platform and precision as its input parameters. Therefore the Backpropagation algorithm did nothing but find this function with a 2% error. There is no noise in the data, for example missing input parameters, missing values, unnecessary parameters or simply wrong values. These are optimal conditions to for a neural network to find a representing function.

The function that PRICE Systems uses to calculate the complexity is quite complex[77]:

[77] Petters, Claus: Computer aided parametric cost estimation, DASA-LA Hamburg, 1995, p. 4-4

$$MCPLXS = \frac{4.3 * (PLTFM^{0.32}) * (NP^{0.4})}{1.35 * (PRECI^{0.081}) * (MI^{0.024})} * (1 + ((N - MA) * 0.06)) + Z;$$

$$N = \begin{cases} 3 \text{ if } \text{PLTFM} < 2 \\ 4 \text{ if } \text{PLTFM} \geq 2 \end{cases};$$

$$Z = \begin{cases} 0 & \text{if } L \leq 1 \\ 0.01 * L^{0.6} & \text{if } L > 1 \end{cases};$$

PLTFM = Platform

PRECI = Precision

NP = Number of parts

MI = Machinability

MA = Maturity

L = Length

Although this function is quite complex a simple neural network trained with some samples only is capable to represent this function. This appears to be a simple way to creating a parametric cost estimating tool. In fact this is not so! The complexity itself gives no statement about the actual cost of a component. It is just an important figure used in the PRICE model to estimate the production costs. Many more parameters are required to calculate the actual costs of a project which is assembled out of many components. According to chapter 3.6.2 production costs can be divided into production costs, prototype costs, development costs and costs of tools and test equipment. PRICE employs these parameters in a mathematical function to determine the final production costs. Therefore a bigger neural network, in terms of more input parameters, should be able to represent this cost estimating function.

5.3 Errors

To develop a mathematical equation PRICE Systems has put together many data from many different projects from many companies into one database. If a company uses this equation to estimate the costs of one project this estimate will automatically be afflicted with an error. This error originates from the fact, that the equation represents only typical values of different companies for the complexity (see chapter 3.6) and it does not consider individual characteristics of the company.

To reduce this error PRICE Systems requires users to calibrate the model by using their own data in order to improve the estimating accuracy of the model for each individual company. Because every company has its own individual style and because every employee has its personal skills every enterprise will have its individual cost characteristics. To adjust the parameters of the estimate it is important to perform a cost analysis on past projects and compare this data with the corresponding estimates.

The calibration method for a neural cost estimate would be different to the calibration of a parametric model. The data from the cost analysis has to be processed so that it can be used as training data for the neural network. For the above mentioned example this would mean that the information from the cost analysis is used to perform a recalculation to gather the actual complexity. Complexity is not an impersonal attribute of an article but rather a particular attribute of an organisation's ability to produce that article. This complexity is combined with the corresponding platform, precision, machinability and maturity of a component. To calibrate the neural network these new recalculated data sets are used as training data for the model. There are two ways to insert this training data:

1. The new model is trained on both data generated by PRICE and from the individual organisation's cost analysis using empirical data. The result would be a function that is mainly based on the mathematical equation of PRICE. This procedure is advisable if the company does not have enough cost analysis data sets at their disposal because then the empirical data of PRICE would help to create a detailed function. In the long-term the model can be successively supplemented with empirical cost analysis data sets in order to calibrate it more and more to the companies structure. The error of the estimate might be a little higher using this method than in the following method, particularly at the beginning, because the individual cost structure of the enterprise is only partly considered.

2. A completely new network is created that exclusively contains the data sets out of the empirical cost analysis only. A prerequisite for this alternative is a sufficient amount of data sets. The advantage of this alternative is, that the calibration error is totally eliminated. The new estimating accuracy might exceed the estimating accuracy of the PRICE model, because the neural network can create a completely new function with an optimal fit to the data set (see chapter 4.2) while the PRICE model has only an predetermined relationship, which can only be modified in a limited way.

5.4 Conclusion

The above mentioned example of a neural model is a new mathematical way to determine the complexity of a component. The stored information inside the network is based on the PRICE model and the mathematical equation and the choice of input parameters (maturity, machinability, platform, etc.) originate from PRICE. It would be easy for a neural net to consider other influencing parameters as long as they are in any functional connection with the output parameter. Then the net would represent a completely new model.

Thanks to available shareware programs it is very simple to program a neural network. Besides NNModel other software programs (e. g. Qnet) have been tested on the same data. Their performance is comparable to the restults of this chapter.

The quality of the represented function with NNModel can be easily tested with test data. Quality in terms of artificial intelligence solely means the characteristic of a functional relationship between input and output parameters. The network does not define the function itself. This is the major disadvantage of neural nets because it is very important for project managers to be able to follow the estimating procedure so that they can have a high level of confidence and are able to persuade others of the accuracy of their estimation. Particularly the Defence Contract Audit Agency (DCAA) demanded in 1980 that financial negotiations for

military orders had to be based on Independent Parametric Cost Estimates (IPCE).
[78] To keep cost estimates comprehensible

- CER's have to be logically set up.
- CER's have to be verifiable.
- there have to be sufficient statistical interdependencies between the parameters.
- the mathematical model has to be verifiable.

For any parametric cost estimate it is easy to trace the determination of production costs because they are based on mathematical equations providing that these relationships are known. On the other hand the project manager might regard a neural network as a mysterious "black box" where some input values result in an output value. This “black box” might be the reason why parametric analysts might be very sceptical about this way of cost estimation unless the project manager is very familiar with neural networks.

To some extent proprietary cost estimating models can be considered as a “grey box” because the complete mathematical basis of the models are not made known to the end user. From the point of view of the manufacturer of such models this is quite understandable as they wish to maintain an element of uniqueness and business secrecy.

Even if it is absolutely necessary for the project manager to have a mathematical equation a neural network can still be of some help. It can support the estimator in developing a mathematical equation. To do so the estimator has to determine the parameters that mainly influence the production costs. The classical procedure then is to collect a sufficient amount of samples and then to perform a regression analysis. Instead of this analysis the estimator can easily develop a neural network with the samples and then check the quality of the network.

---

[78] Starrett, Charles O., Jr.: Parametric Cost Estimating - An Audit Perspective, In: ISPA Journal of Parametrics, Vol. I, No. 4, Spring 1982, p. 3

If the neural network has a bad quality this simply means that there is no functional connection between input and output parameters. Then the estimator has to determine other parameters and start all over again.

If the neural network finds any functional relations in the data the estimator can now perform the regression analysis to find the underlying relationships. Although the relationship is not made explicit by the neural network some software programs give an indication about the strength of the influence of different parameters to the ultimate output.

The advantage of this procedure is that the estimator can easily use a neural network to decide whether it is worthwhile performing a regression analysis. A simple neural network helps to determine if a functional connection exists. The major disadvantage is that the multiple regression analysis, which is relatively simple to apply, is only able to develop a hyper plane as was outlined in a previous section of this chapter. This means that even if the neural network finds functional interdependencies between input and output parameters a multiple regression analysis might not be capable of finding a function that represents the data as well as the neural network. A non-linear regression analysis on the other hand is capable of representing the data in all dimensions but because it is based on trial and error it can be very time consuming.

6 Multi-Layer Perceptron to determine Costs

In the previous chapter a neural network was established to represent an equation which calculates the MCPLXS according to the PRICE model (see chapter 5). With the knowledge that the complexity is based on an equation that uses machinability, maturity, platform and precision as input parameters a quite limited amount of about 200 data sets as training data is sufficient to derive the function, because the results of the network could be easily checked with the equation (see chapter 5.2). Establishing a neural network to calculate the MCPLXS is the first step to set up a cost estimating model according to PRICE. The next question is whether a neural model is able to calculate a cost estimate that is based on the PRICE model.

As mentioned in chapter 5 the main parameters required to estimate the costs of a component are complexity and weight. Next to the these PRICE uses other cost influencing parameters like quantity, volume and production period, which are now carefully explained:

- **Quantity (QTY):** The quantity is the number of parts that are going to be produced. The greater the number produced the greater the total cost of the project. Assuming that in a production run the unit costs diminish as the number produced increases because of the influence of learning it can be stated, that the shape of the quantity/cost-function is not a linear but a diminishing function.

- **Weight (WT):** The weight is probably the major influencing parameter because according to the PRICE model the production costs of a component are mainly determined by complexity and weight (see chapter 5). The heavier a component the more material is required and the more expensive it is expected to be. PRICE distinguishes between the weight of electronic components which have a considerably low weight but still a high influence on the costs and the

structure weight which describes the mechanical components. Together they result in the total weight.

- **Volume:** Often there is a 100% correlation between volume and weight (if a component is twice as big it is usually twice as heavy, too). Because a neural network requires input parameters that are independent from each other in the following example this value will not be considered. There are only a very few cases in which the volume is not proportional to the weight; electronic parts might be a case. A big tower computer case for instance can contain only very few electronic parts and therefore be quite light while a laptop is relatively heavy since all necessary components are stuffed together as close as possible. In these cases the volume must be used as extra parameter in the network. PRICE uses a wecf-factor (weight of electronic parts per cubic foot) to consider this phenomena.

- **Project start date (PSTART):** Month and year when the project starts.

- **Project end date (PEND):** Month and year when the project is expected to be finished. PSTART and PEND determine the duration of a project. The shorter the period in which a component is expected to be built the more expensive the project will be because a higher performance regarding the organisation and production is required. If PEND is set to 0 PRICE automatically calculates the optimal duration of the project from the economic point of view.

- **Complexity (MCPLXS):** This is the manufacturing complexity and was described in chapter 5.

- **Production cost:** The final estimated production cost.

Originally using data from RCA, by whom PRICE was initially developed, and later from using extensive data provided by collaboration with many different enterprises, PRICE Systems developed cost estimating relationships and a broader cost estimating system. PRICE is now used world-wide to calculate the costs of

QTY	WT	PSTART	PEND	MCPLXS	Prod. Cost
1	1	196	1296	4.518	492.27
1	1	196	1297	4.568	521.74
1	1	196	597	4.518	525.24
1	1	196	598	4.518	528.79
1	1	196	899	4.518	533.01
1	1	196	100	4.518	534.39
1	4.5	196	1297	3.582	656.62
1	1.5	196	1296	4.518	670.29
1	1.5	196	597	4.518	720.12
1	1.5	196	598	4.518	724.98
1	1.5	196	899	4.518	730.76
1	1.5	196	100	4.518	732.65
1	2	196	1296	4.518	834.61
1	2	196	597	4.518	900.82
1	2	196	598	4.518	906.89
1	2	196	899	4.518	914.12
1	2	196	100	4.518	916.49
1	2.5	196	1296	4.518	989.48
1	2.5	196	597	4.518	1071.66
1	2.5	196	598	4.518	1078.87
1	2.5	196	899	4.518	1087.48
1	2.5	196	100	4.518	1090.29
1	1	196	1296	5.5	1128.19
1	3	196	1296	4.518	1137.22
1	3	196	597	4.518	1235.03
1	3	196	598	4.518	1243.34
1	3	196	899	4.518	1253.25
1	3	196	100	4.518	1256.49
1	3.5	196	1296	4.518	1279.29

Table 10: Generated Production Cost with PRICE

hardware components. Due to the continuous change of the production process and economic circumstances the cost functions used have been refined to improve the estimating precision possible. Part of the international success of PRICE-systems is based on this "secret" formula. It is quite understandable that for commercial reasons this equation is not made accessible to the users of the system.

Even so it would be very interesting to find out, if a neural network is able to represent this unknown function. Unlike the exercise carried out in chapter 5 it is not possible to verify this new network with the help of a known equation and therefore much more data is required to set up and check the neural network. Daimler-Benz Aerospace Airbus Industries (DASA) in Hamburg provided about 6,000 data sets which have been generated with the PRICE-Model for this purpose and Table 10 represents a sample of the available data.

Within the following chapters a fairly high number of analyses have been carried out which are based on this data. A complete print out of these calculations would cover several hundred pages and therefore only data samples, significant values or the results of the following analyses will be displayed. Still to give the reader the opportunity to look at all data sets a reference floppy disc with a compressed self-extracting Excel 97 spreadsheet has been included in this thesis with all statistical analyses on it.

Data for the following quantities of production were provided: 1, 5, 10, 20, 50, 80, 100, 200, 500, 800, 1000, 1500, 2000, 2500, 3000, 4000, 5000, 6000, 8000.

For each quantity the following weights were used: 0, 1, 1.5, 2, 2.5, 3, ... , 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100, 120, 140, 160, 180, 200, 300, 400, 500, 600, 700, 1000, 1800, 1900.

The project start date was mainly 01.96 but there are also samples for 01.93, 01.94 and 01.95. For those values where the project start date is 01.96 the following project end dates have been chosen: 0, 12.1996, 05.1997, 12.1997, 04.1998, 05.1998, 12.1998, 08.1999, 01.2000, 01.2002.

The complexity has 128 different values between 3.518 and 8.069. Most of these values are applied about 20 times. The following complexities are more frequently applied because they are commonly used in the aerospace industry: 4.518 (1809x), 5.5 (441x) and 5.85 (630x).

6.1 Qnet

Due to the fact that it is not known whether the PRICE generator is based on a mathematical formula or on a database or on both, the software program Qnet version 2.1 is used because it has much more powerful tools to analyse the network (see chapter 4.3) than NNMODEL. The main characteristics of the Qnet control panel (see Figure 27) will be described in the following chapters.

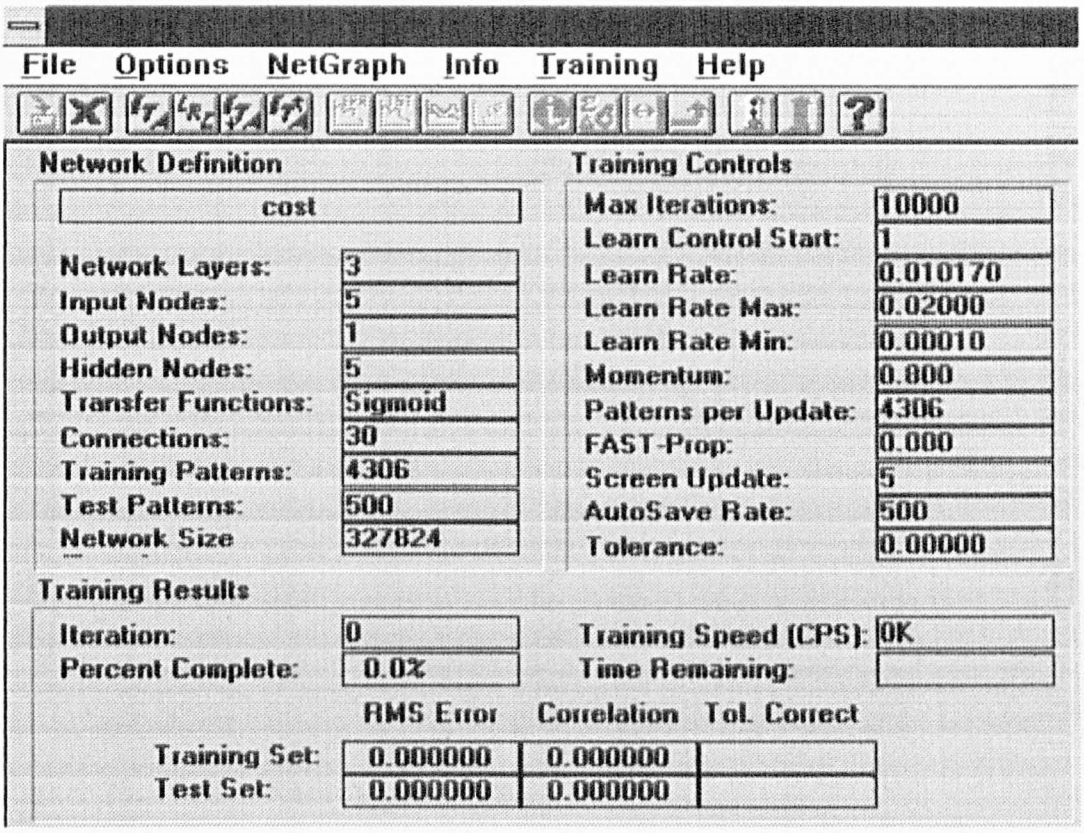


Figure 27: Qnet control panel

6.1.1 Hidden Layer

As mentioned in previous chapters 3 network layers (input, hidden and output layer) are sufficient to represent any function. In this example there are five input nodes (see Table 10) to determine one output node (cost). Again the question is how many neurons in the hidden layer are necessary to represent the requested function? A very convenient feature of Qnet is that it gives the user the opportunity to check how the hidden nodes are being utilised by the network. For networks that are over designed in the hidden layer structure, many nodes may contribute little or nothing to the output response. The "Hidden Node Analyser Plot" shows the nodes' percentage contribution to the layer's output signals over all training patterns. If there are many nodes in the hidden layer that are showing limited contributions, then this layer probably has too many nodes. Likewise, if all nodes show strong contributions then it is possible the adding extra nodes would help the model. This way the ideal number of hidden nodes can be found. To calculate the costs with the above mentioned data 7 networks with 3 to 9 neurons

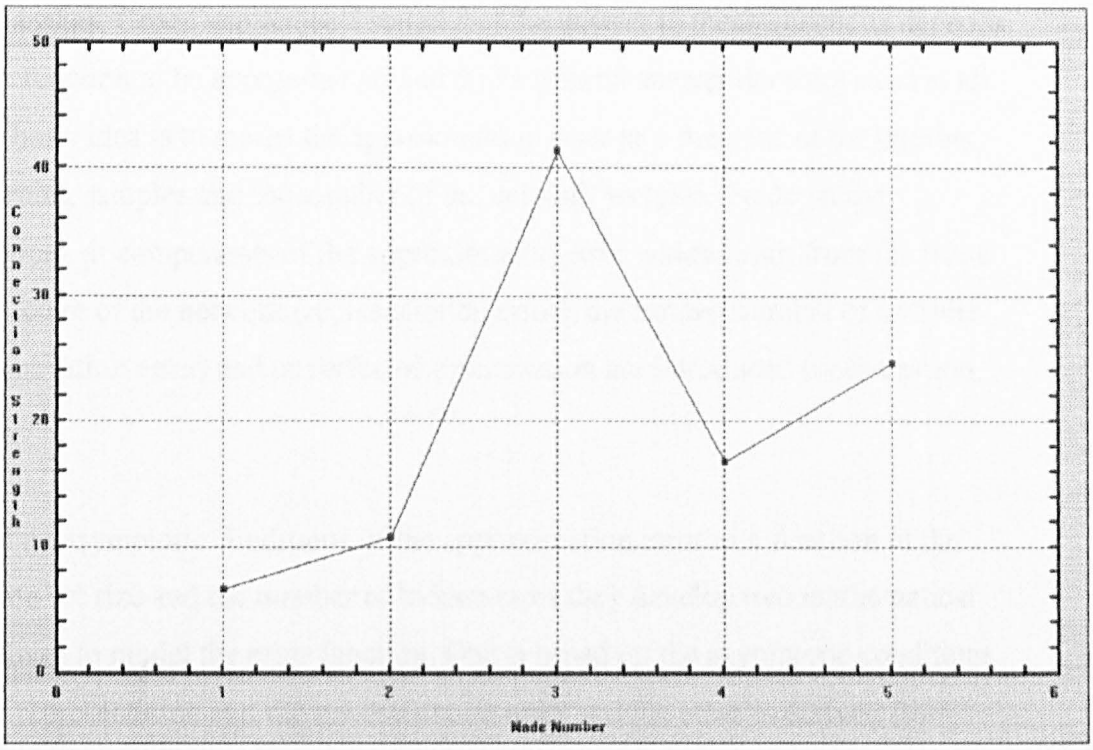


Figure 28: Hidden Node Analyser Plot

in the hidden layer were tested. Figure 28 shows the connection strength for the network with 5 neurons in the hidden layer. Because node number 1 has a quite low influence on the output (about 6%) even a network with 4 nodes only could be sufficient to represent the data. But to be on the safe side the data will be used on a network with 5 neurons in the hidden layer.

6.1.2 The optimal number of learning samples and hidden units

Some software programs help the user to find the right amount of nodes in the hidden unit (e.g. Qnet see chapter 6.1.1) by displaying to which extent each hidden unit influences the output units. This is a practical way to design an adequate model. If some hidden neurons have hardly any influence on the output they can simply be erased without significantly changing the behaviour of the net. The question arises whether an analytical method exists to determine the optimal number of hidden units. Another question which occurs is how many learning samples are necessary to train a neural net? If answers to these problems exist the training process of neural nets could be significantly reduced.

Vysniauskas, Groen and Kröse[79] stated that the answer to these questions depends on the function to be approximated and that a general answer does not exist at all. Their basic idea is to model the approximation error as a function of the number of learning samples and the number of the network weights. To do so the definitions of components of the approximation error which result from the finite architecture of the network (representation error), the limited number of samples (generalisation error) and unperfected optimisation are introduced (optimisation error).

Based on asymptotic conditions of the approximation error as a function of the learning set size and the number of hidden units they develop two mathematical guidelines to model the error function. One is based on the asymptotic conditions of the representation and the generalisation error and the other is derived from Barron's formula[80]. An important contribution about the approximation capabilities of feedforward networks was made by Barron who pointed out that for some classes of smooth functions the mean integrated squared error between the estimated network and the target function is bounded by $O(1/h)+O(h/N)\log N$ where h is the number of hidden units and N is the number of training examples and O is the output.

Both models are mathematically complex and require test data regarding the set-up of different neural networks, which can itself be quite time consuming. Barron reports that when applied on an example the predictions of the errors of both models differed about 25% (the predicted error has an error of 25%). Barron also says that the influence of possible noise in the learning samples in the model has been neglected and needs some further attention in further investigations.

---

[79] Vytautas Vysniauskas, Frans C. A. Groen, J. A. Kröse: The optimal number of learning samples and hidden units in function approximation with a feedforward network, Technical Report CS-93-15, University of Amsterdam, Faculty of Computer Science and Mathematics, The Netherlands, 1993

[80] Barron, A., R.: Approximation and estimation bounds for artificial neural networks, Proceedings of the Fourth Annual Workshop on Computational Learning Theory, 1991, pp 243-249

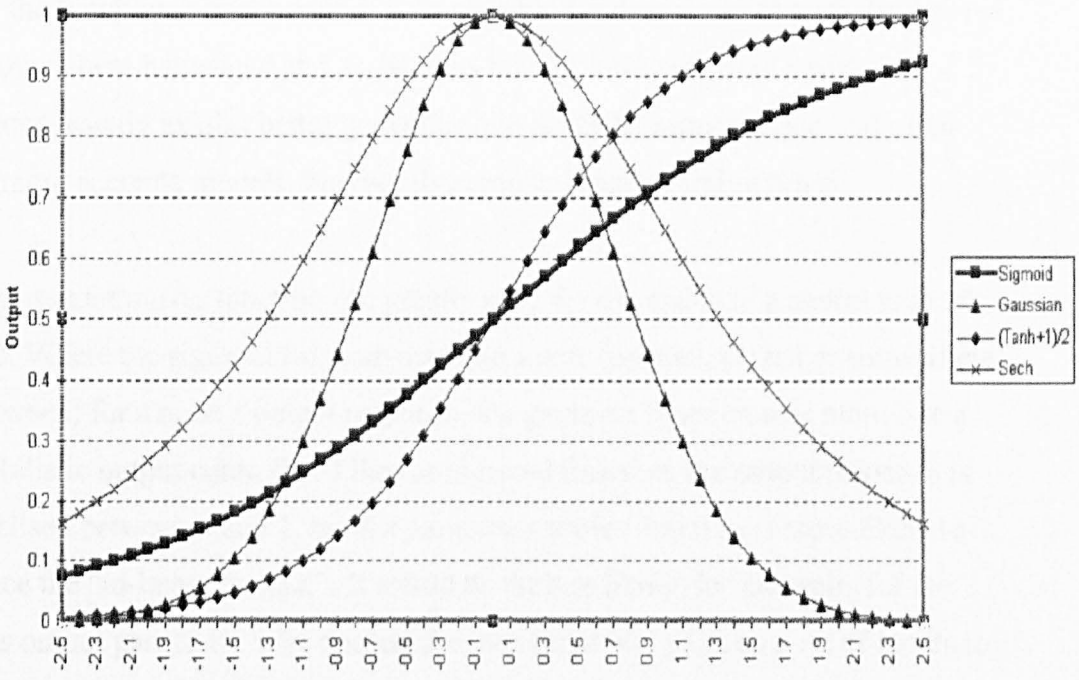


Figure 29: Types of Transfer Functions[81]

These reasons led to the author of this thesis' conclusion that although the approach of Vysniauskas, Groen and Kröse demonstrates a very interesting way to determine the optimal number of learning samples and hidden units this method is not applicable to the data provided by DASA.

6.1.3 Transfer Function

A node's transfer functions serves the purpose of controlling the output signal strength for the node (see chapter 2.3). These functions set the output signal strength between 0. and 1. The input to the transfer function is the dot product of all the node's input signals and the node's weight vector. Qnet gives the option of selecting two distinct types of transfer functions: the sigmoid and the gaussian.. Figure 29 shows the behaviour of each function.

This sigmoid function is the most widely used function for backpropagation neural networks. It is represented by the mathematical relationship $1/(1+e^{-x})$. The sigmoid function acts like an output gate that can be opened (1) or closed (0).

[81] Help Manual of Qnet ver. 2.1: Neural Network Training - Learning Modes, Vesta Services, Inc. 1994, 1995

Since the function is continuous, it also possible for the gate to be partially opened (i.e. somewhere between 0 and 1). Models incorporating sigmoid transfer functions usually exhibit better generalisation in the learning process and often yield more accurate models, but can also require longer training times.

The gaussian transfer function can greatly alter the dynamics of a neural network model. Where the sigmoid function acts like a gate (opened, closed or somewhere in-between) for a node's output response, the gaussian function acts more like a probabilistic output controller. Like the sigmoid function, the output response is normalised between 0 and 1, but the gaussian transfer function is more likely to produce the "in-between state". It would be far less likely, for example, for the node's output gate to be fully opened (i.e. an output of 1). Given a set of inputs to a node, the output will normally be some type of partial response. That is the output gate will open up partially. Gaussian based networks tend to learn quicker than sigmoid counterparts, but also tend to produce networks that are prone to memorisation with less generalised learning.

The hyperbolic function counterparts to the sigmoid and gaussian functions are the hyperbolic tangent and hyperbolic secant functions. The hyperbolic tangent is similar to the sigmoid but can exhibit different learning dynamics during training. It can accelerate learning for some models, but it also may not achieve the same accuracy as a sigmoid based models. Experimenting with different transfer functions with a particular model is the only way to conclusively determine if any of the non-sigmoid transfer functions will improve learning characteristics.

For the vast majority of network designs (including the network that is based on the data of chapter 5) the sigmoid function performs best. A general rule of thumb (but not always the case), is that the sigmoid will produce the most accurate model and, likely, the slowest learning. If it is intended to frequently train similar models and/or if training speeds is important, it might be advisable to experiment with different combinations of transfer functions, including hybrid networks, in search of the fastest training models that produce acceptable accuracy.[81]

6.1.4 Learning Rates and Learn Rate Control (LRC)

The backpropagation training paradigm uses two controllable factors that affect the algorithm's rate of learning. To optimise the rate at which a network learns, these factors must be adjusted properly during the training process. The two factors are the learning rate coefficient, α, and the momentum factor μ (see chapter 2.6.4.5). The valid range for both α and μ is between 0 and 1. Higher values adjust node weights in greater increments, increasing the rate at which the network attempts to converge, while lower values decrease the rate of learning. Just as there are limits to how fast a brain can learn ideas and concepts, there are also limits to the rate at which a network can learn. If a network is forced to learn at a rate that is too fast, instabilities develop that can lead to a complete divergence of the training process.

The learn rate coefficient can be controlled manually during training or Qnet can control it automatically using its Learn Rate Control (LRC) feature. LRC will drive α higher or lower in a systematic fashion depending on the current learning activity. If the network appears to be learning at a relatively slow rate, α is driven up quickly. Conversely, if the network is learning at a fast pace, Qnet will raise α only slightly, hold it constant, or even lower it to avoid instabilities. If at any time the network shows signs of instability (seen as oscillations in the training error), α is lowered quickly to damp the instabilities. Damping instabilities is critical to preventing complete training divergence. The LRC feature can be turned on and off interactively during the training process, and it can be activated at set-up time by specifying the iteration number that LRC will start.

Occasional interaction with the LRC system can help to improve the learning process. For example, let's assume that a network is training with LRC active. After several hundred iterations NetGraph is used to view the α history. The graph shows that whenever α exceeds a value of 0.15, instabilities occur (seen as oscillations in the RMS error) and α is dropped substantially to avoid divergence. During each recovery process, learning slows due to lower learning rates. By setting appropriate minimum and maximum values for α, LRC will keep α below

the established upper limit and above the specified minimum. For the above example, α could be limited to a maximum of 0.12 to prevent the instabilities from occurring. This will keep the network learning at an optimal pace by preventing the slow downs required to recover from instabilities. It is common for these limits to change gradually over many iterations (usually the upper limit decreases during the training process, but not always). Repeat this procedure when instabilities develop on a regular basis.

LRC concerns itself only with control of α. Usually, little or no interaction is required with the momentum factor (μ). The momentum factor damps high frequency weight changes and helps with overall algorithm stability, while still promoting fast learning. For the majority of networks, μ can be set in the 0.8 to 0.9 range and left there. However, there is no definitive rule regarding μ. Some networks may train better with μ values set at a lower level. Some networks train perfectly with no μ term used at all (set to 0). Most neural modellers prefer to use higher momentum values, since this usually has a positive effect on training. If training problems occur with a given μ value, it may be helpful to experiment with different values. μ can be changed interactively at any time during the training process with Qnet.

Note: For the vast majority of networks, LRC is an effective tool preventing divergence and keeping α in a range that will improve learning speeds. If a model exhibits poor learning characteristics with LRC active (i.e. training divergence or many instabilities), simply turn LRC off (Options menu of the training window) and set α to a value low enough to guarantee stable learning. Networks employing gaussian transfer functions may find that LRC is less effective in preventing divergence in some cases.

When all training cases are not used in each weight update cycle (Patterns per Weight Update Cycle), LRC is not recommended. Error descent can be somewhat noisy and/or training characteristics can be adversely affected by varying the learn rate. Taking manual control of the learning rate for such models is recommended.

6.1.5 Learning Modes

An important concept to understand about the training process is that there are two distinct types of learning that can take place. One type is generalised learning where the network develops an understanding of how the inputs can best be generalised to formulate an output prediction. The other type is "memorisation" where the network can, in effect, recall a set of outputs after being presented with the inputs. An example of generalised learning is where a person studies how to add together a small set of numbers (i.e., 2+2, 3+5, etc.) and through understanding some basic concepts, that person can determine the results of any two numbers presented to him (even if they were not previously studied). An example of memorisation learning is where a person learns the US state capitals. Learning the capitals of 45 states does little to help a person predict what the other 5 might be. Memorisation learning is only useful for the learned set. It offers no help in determining solutions outside the learned set. Differentiating between the learning modes will allow to optimise model training and better determine the effectiveness of the a model prior to practical use. [81]

Backpropagation attempts to drive the network's response error for the training set to a minimum value. The error value monitored during Qnet training is the root-mean-square (RMS) error between the network's output response and the training targets (equivalent to the standard deviation). When the training set's error is descending during the training process, one or both types of learning discussed above is taking place. Unfortunately, there is no way to determine which type of learning is taking place by monitoring the training set error by itself. To determine the type of learning, a test set (or overtraining set) must be employed. Qnet allows the test set to be monitored interactively during training. This set of data is not used to train the network, however, the error in the network response is monitored to determine how the network responds to patterns outside the training set. If both the training and test set errors are declining (see first example in Figure 30), cognisant learning predominates since the network is learning to generalise the relationships between the inputs and outputs. When a test set's error has reached a minimum level and begins to increase indefinitely thereafter, overtraining is occurring (see third example in Figure 30).

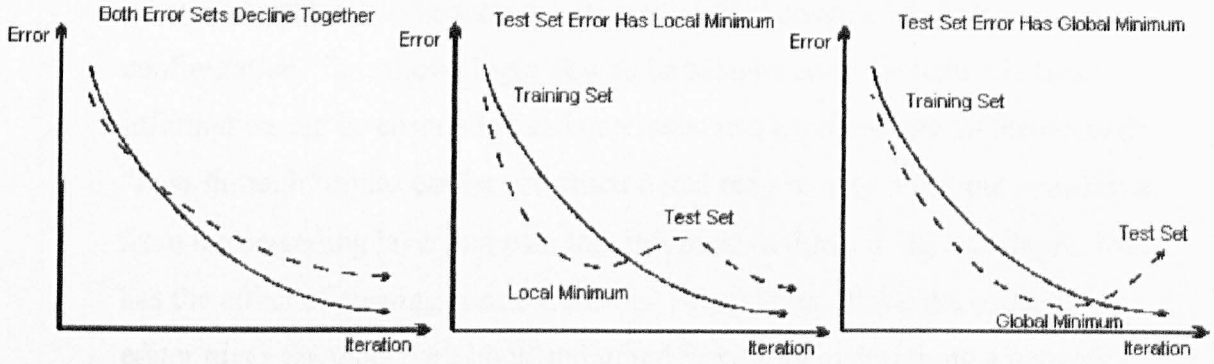


Figure 30: Training set and Test set errors

Another possibility is that some networks may exhibit long periods of training where the test set error increases before declining again (see second example in Figure 30).

If a test set's error begins to increase, training should be continued to determine whether the minimum is local or global in nature. The AutoSave feature of Qnet allows to return to a point at or near the minimum if it is global in nature.

It should be noted that the method of determining the learning modes and overtraining status by monitoring the training and test set errors assumes that the test set is an adequate subset of the training set. This may not always be true. For problems where the test set is some limited or organised subset of the training set cases, the minimum test set error may simply indicate the point that the network has best modelled that subset of test set cases. To function as a true overtraining indicator, it is important that the test set cases are a truly random and broad sample of the training cases. Qnet may be instructed to select the cases randomly from the training set. While this does not guarantee a perfect set for testing, a high probability will exist that an ample amount of unique case types will be present when large test sets are employed. [81]

6.1.6 Other Network Definitions

- **Network Connections**: Another network design consideration concerns how to control the network's connections. Qnet implements a connection editor that

allows connections to be removed from the fully connected default configuration. This allows logic flow to be introduced to the network. Input information can be channelled and processed in a localised area of the network. "Pass-through" nodes can be constructed that receive only one input connection from the preceding layer and pass that information down to the next layer. This has the effect of creating connections that skip a layer. While the connection editor gives the modeller almost unlimited flexibility in designing a network, the fact is that the vast majority of designs work best fully connected. Therefore in this example a fully connected network has been chosen. The connection editor is best suited for highly advanced models that require groups of input data to be processed through separate network pathways.

- **Training Patterns**: Amount of data sets used to train the network.

- **Test Patterns**: Amount of data sets used to test the network.

- **Network Size**: Size of network in Bytes

- **Max Iterations**: Maximum number of iteration steps to train the model

- **Learn Control Start**: Iteration number to begin Learn Rate Control. During the initial training iterations (50 to 100) for a new network, LRC should normally be turned off. The node weights of new networks are adjusted rapidly during initial iterations and the training error can oscillate wildly. Using LRC during this period is perfectly safe, however, the LRC algorithm may drive α unnecessarily low in an attempt to eliminate these normal oscillations. [81]

- **Learn Rate**: The actual learn rate during the learning process.

- **Learn Rate Max**: The maximum learning rate to be used during Learn Rate Control.

- **Learn Rate Min**: The minimum learning rate to be used during Learn Rate Control.

- **Momentum**: The momentum factor μ (see chapter 2.6.4.5)

- **Patterns per update**: The number of patterns to process per weight update cycle can have a large effect on the overall training process and convergent behaviour. Qnets default is to process all training patterns prior to updating network weights. This allows a global error vector to be developed prior to adjusting weights. This practice generally leads to the most orderly decent of both training and test set errors at the price of slightly slower training performance. This method, however, is strongly recommended for training sets where non-precise and somewhat noisy relationships exist between the inputs and outputs. [81]

- **FAST-Prop**: The FAST-Prop coefficient controls the algorithm used by Qnet for training. FAST-Prop training can accelerate training for some networks. If the FAST-Prop coefficient is set to 0 (the default), Qnet will employ its backpropagation algorithm to train the network. If the coefficient is set to a value above 0.0 (to a maximum of 3.0), the FAST-Prop algorithm is used. The closer the coefficient is set to 0.0, the closer FAST-Prop approximates standard backpropagation. While the FAST-Prop training method can often accelerate the learning process, a drawback with this method is that there is a risk that this algorithm will not converge to a minimum error, especially when higher coefficient values are used. For this reason, in this example the standard backpropagation method is used. [81]

- **Screen update**: Iteration interval at which screen updates occur during the training process.

- **Auto Save Rate**: The AutoSave feature of Qnet allows to recover from overtraining situations and training oscillation. The user specifies a rate (or interval) at which the network should be stored. For overtraining, return to the

first stored iteration prior to the point that the test set error started to increase. To recover from an oscillating network, return to the first iteration prior to the oscillation. [81]

- **Tolerance**: Selects the tolerance to used to monitor training accuracy.

6.2 Test 1

6.2.1 Preparation

During some initial exploratory tests on the PRICE data provided by DASA the author became aware that PSTART and PEND do not contribute any useful information to the network in finding the right function. The reason for this can be explained as following:

PSTART and PEND together indicate the duration of the project (duration = PEND-PSTART). By looking at data records where all values but PEND and production cost are equal, it can be said that the longer a project lasts the more expensive it will be. Whether this statement is true or not should not be discussed in this chapter since the data is just referring to an existing model.

PSTART or PEND indicate in which month and year the project will start and finish. By looking at comparable data records the PRICE model considers annual inflation rates for the years during which the project will be executed. The estimate of the same project at a later period of time is more expensive than an earlier project. According to the sales department of PRICE Systems a cost index (comparable to Financial Times Index, Aerospace or Electronics Cost Index) is implemented to take inflation into consideration.

If month and year of PSTART or PEND (e.g. 797 for July 1997) are used as input values the neural network might not be able to interpret any duration or moment out of the data. For example the duration of a project that starts in December 1996 (1296) and ends in June 97 (697) would be negative (-599) while another project

with PSTART = 497 and PEND = 498 would have the duration 1. To make a comparison between the data possible PSTART and PEND are transformed into numerical values where 1 represents the 1st of January 1900 and for each following day 1 is added, i. e. the starting date of the 1st December 1996 is 35065. PEND is erased from the data and instead a column with the calculated duration is added to the data record, as is shown in Table 11.

Weight, MCPLXS, PSTART, Duration, Quantity are input parameters. Prod.Cost is the output value provided by PRICE. There is one hidden layer with 5 hidden neurons (see chapter 6.1.1). The network is fully connected. 500 data records were randomly excluded from the complete file to provide the test data; the remaining data were used as the training set.

6.2.2 Results

The results of this network are much less impressive than for the exercise carried out in chapter 5.2 where the complexity has been generated from a known function. Applying the same formulas as in chapter 5.2 the relative mean error of the predicted values of the test set is **38,638%** (absolute mean error: 13,640,683.93). The maximum error is 218,293,500%.

6.2.2.1 Normalisation with rounding errors

Table 11 shows the input parameters and network results of the first 22 data records (those marked with an asterisk were randomly excluded from the training set and used as test values). The outputs represent the production costs calculated by the network while the targets are the renormalised Prod.Cost which the network is aiming at during the training algorithm. The targets are the renormalised production costs and should equal the production costs but in fact there is a difference. The reason for this is the normalisation (see chapter 2.9). Backpropagation neural networks require that all training targets are normalised between 0 and 1 (-1 and 1) for training. This is because an output node's signal is restricted to a 0 to 1 (-1 to 1) range.

No.	WT	MCPLXS	PSTART	Duration	QTY	Prod.Cost	target	output
1	1	4.518	35065	335	1	492.27	523.33	-16,931,000.00
2	1	4.568	35065	700	1	521.74	523.33	-16,751,000.00
3	1	4.518	35065	486	1	525.24	523.33	-16,875,000.00
4	1	4.518	35065	851	1	528.79	523.33	-16,740,000.00
5	1	4.518	35065	1308	1	533.01	523.33	-16,573,000.00
6	1	4.518	35065	1461	1	534.39	523.33	-16,517,000.00
7	4.5	3.582	35065	700	1	656.62	678.65	-17,024,000.00
8	1.5	4.518	35065	335	1	670.29	678.65	-16,867,000.00
9	1.5	4.518	35065	486	1	720.12	756.31	-16,810,000.00
10	1.5	4.518	35065	851	1	724.98	756.31	-16,676,000.00
11*	1.5	4.518	35065	1308	1	730.76	756.31	-16,509,000.00
12	1.5	4.518	35065	1461	1	732.65	756.31	-16,452,000.00
13	2	4.518	35065	335	1	834.61	833.97	-16,802,000.00
14	2	4.518	35065	486	1	900.82	911.63	-16,746,000.00
15	2	4.518	35065	851	1	906.89	911.63	-16,611,000.00
16	2	4.518	35065	1308	1	914.12	911.63	-16,443,000.00
17	2	4.518	35065	1461	1	916.49	911.63	-16,387,000.00
18	2.5	4.518	35065	335	1	989.48	989.29	-16,737,000.00
19	2.5	4.518	35065	486	1	1071.66	1,066.90	-16,682,000.00
20	2.5	4.518	35065	851	1	1078.87	1,066.90	-16,547,000.00
21*	2.5	4.518	35065	1308	1	1087.48	1,066.90	-16,379,000.00
22	2.5	4.518	35065	1461	1	1090.29	1,066.90	-16,322,000.00

Table 11: Results 1 - beginning of data record

Since the applied data has values up to 3,648,143,574 the normalised numbers should have at least 10 digits after the decimal point in order to calculate precisely. When Qnet scales the data down to the 0 to 1 range the software uses numbers with only up to 5 digits after the decimal point for programming reasons. Therefore the production costs are subject to rounding errors which are relatively high particularly for small numbers. According to the user manual Qnet uses floating point numbers during the learning algorithm. Therefore no rounding error occurs during the training phase.

During renormalisation the computer can not distinguish between relatively small values. For example the first six data sets of Table 11 all have different Prod.Cost values but all have identical target values of 523.33. This is because the Qnet Software looses precision due to rounding of small numbers.

6.2.2.2 Training process

The discrepancy between targets and outputs is also very high for small Prod.Cost (see Table 11). On the other hand it is very striking that predicted costs (output)

No.	Prod.Cost	Target	Output	error
6,205	2,505,293,767	2,505,300,000	2,404,100,000	-4.04%
6,206	2,522,015,011	2,522,000,000	2,416,900,000	-4.17%
6,207	2,876,071,858	2,876,100,000	2,797,300,000	-2.74%
6,208	2,895,281,077	2,895,300,000	2,810,100,000	-2.94%
6209*	3,571,868,893	3,571,900,000	3,453,200,000	-3.32%
6,210	3,595,752,807	3,595,800,000	3,463,600,000	-3.68%
6,211	3,648,143,574	3,648,100,000	3,490,000,000	-4.33%

Table 12: results 1 - error of last data records

and actual costs (target) for the last data records (see Table 12) almost match. The reason for this can be found in the backpropagation training algorithm.

In the data record the production costs are spread over several orders of magnitude (from 492.27 to 3,490,000,000). Before the backpropagation training is started, all weights of the network are randomly chosen within a certain range. This usually leads to constant net outputs that are quite independent from the applied inputs (see Figure 31). The test outputs (blue) can hardly bee seen in this figure because they almost match with the training outputs (green).

According to equation 4 in chapter 2.6.4.3 the network error is calculated as the sum total of the absolute square errors ($E = \frac{1}{2} * \Sigma_i (a_i - o_i)^2$). The square in this formula causes a exponential influence of high errors which particularly occur at the end of this data record where output values are extremely high. During Backpropagation training the adjustment of weights is chosen in such a way that small changes lead to maximum minimisation of the error (gradient descending method). This virtually means that the neural net mainly concentrates its training on the last few records (because the error is very high there) while small values are almost neglected.

After a certain amount of training cycles when the "big" errors have been reduced the Backpropagation can pay some more attention to the smaller values. But only minimal weight changes are allowed because at the same time they immensely influence the error at the steep courses of the cost curve. This conflict makes the

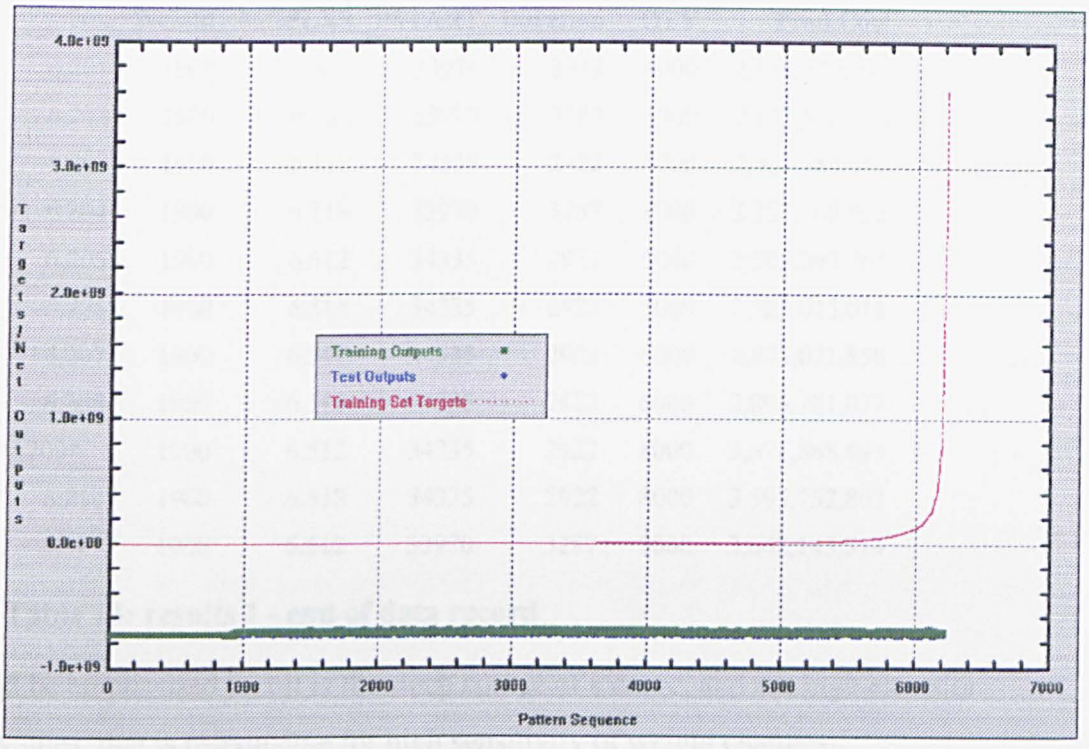


Figure 31: net outputs before training

training process very slow and hardly leaves any chance for smaller values to adapt.

6.2.2.3 Steep cost curve

The red line in Figure 31 displays the target costs for each pattern. It is very striking that there is a vast increase of costs for the patterns larger than 5000. The data appears to follow an exponential curve. Only little changes of the input data lead to enormous changes of the output. For example, if pattern 6,207 and 6,208 shown in Table 13 are compared it can be observed that a small difference of the complexities of 0.006 rises the costs by 19,209,219 DM. Pattern 6,203 and 6,204 show that a project starting in January 94 and ending in January 2002 (PSTART = 34335, Duration = 2922) can be 48,002,292 DM cheaper if you finish in the same year but start one year earlier (January 93 = PSTART = 33970, Duration = 3287).

The steep course of the Training Set Targets curve in Figure 31 makes it difficult for Backpropagation training to find an appropriate adjustment of the weights of the synapses in the neural net (small changes have a great influence on costs). It

No.	Weight	MCPLXS	PSTART	Duration	QTY	Prod.Cost
6,201	1900	6.512	33970	3287	4000	2,158,450,050
6,202	1900	6.518	33970	3287	4000	2,172,873,633
6,203	1900	6.118	34335	2922	8000	2,304,384,610
6,204	1900	6.118	33970	3287	8000	2,352,386,902
6,205	1900	6.512	34335	2922	5000	2,505,293,767
6,206	1900	6.518	34335	2922	5000	2,522,015,011
6,207	1900	6.512	34335	2922	6000	2,876,071,858
6,208	1900	6.518	34335	2922	6000	2,895,281,077
6209*	1900	6.512	34335	2922	8000	3,571,868,893
6,210	1900	6.518	34335	2922	8000	3,595,752,807
6,211	1900	6.512	33970	3287	8000	3,648,143,574

Table 13: results 1 - end of data record

should be emphasised that it is the steep course of a curve, and not high errors or high values, that is responsible for high sensitivity of weight changes.

Generally the validity of the PRICE model should be questioned for values above 15,000,000 DM. According to Mr. Hübner, engineering consultant in the department for parametric cost analysis of Daimler-Benz Aerospace Airbus GmbH (DASA) in Hamburg, these values have only been generated by the PRICE-Model and never been verified because the majority of all cost estimates at DASA lie within 5,000.-DM and 300,000.-DM (Deutsche Mark). Therefore they have no practical relevance at all[82].

For the calculation of complicated and therefore expensive systems (e.g. aeroplanes, space shuttle etc.) this mathematical model is not directly used. The system is rather divided into smaller subsections (e.g. wings, engine, landing gear, fuselage etc.) which again can be broken down to smaller elements and then for each of these elements a cost estimate according to the model is carried out. It is very unlikely that any components exceed the amount of 15,000,000 DM. Therefore it can be assumed that the PRICE model supplies the user with acceptable results within a certain cost interval. Values outside this range can be mathematically determined but they have no practical use.

[82] Telephone call between Mr. Hübner and author on February, 27th 1997 14.30

6.2.3 Conclusion

As discussed in the previous chapter, there are mainly three reasons that are responsible for the poor performance of the neural net:

1. An extremely steep course of the cost curve for high values makes it difficult during Backpropagation training to adjust the weights adequately. A possible approach is to ignore inputs with Prod.Cost greater than 15,000,000 DM.
2. High errors at the end of the data record cause small output values to be neglected during the training process. Again a possible approach is to ignore inputs with Prod.Cost greater than 15,000,000 DM and/or apply logarithmic transformation to Prod.Cost.
3. A widely spread data range leads to normalisation with rounding errors. A possible solution is to apply a logarithmic transformation to Prod.Cost.

At the end of this evaluation 126 data records, which have their duration value set to 0, have been erased out of the file. PRICE Systems uses this value to calculate the optimal duration of a project. For network training this value is most unsuitable because it does not fit among the other data. Despite of the poor performance of this neural net, these data records still have been recognised as outliers.

6.3 Test 2

6.3.1 Preparation

The preparation of data and network is identical to chapter 6.2.1 (Test 1). The data set has been modified in accordance to the conclusions of chapter 6.2.3. All data records which have Prod.Cost greater than 15,000,000 have been excluded and Prod.Cost values are transformed according to the following formula:

$$\text{Prod.LogE} = \log_e \text{Prod.Cost}$$

target Prod.Log.E	Output Prod.LogE	Error	Error [%]	target Prod.Cost	ouput Prod.Cost	Error	Error [%]
7	9	2	29%	1,097	8,103	7,006	639%
21	23	2	10%	1,318,815,734	9,744,803,446	8,425,987,712	639%

Table 14: example for influence of logarithmic transformation

With this transformation the output is in a range from 6.199027346 to 16.5210993. The logarithm ensures that both among the smaller values and among the large values a distinction is possible. The natural logarithm of the Prod.Cost has been used so that there is a reasonable range of the values of Prod.LogE.

6.3.2 Results

The mean error of the predicted values of the test set is 78% (the absolute mean error is 887,038.5). The maximum error is 1,400%. Compared to the results in Test 1 of chapter 6.2 (mean: 38,638%; max.: 218,293,500%) these new errors are just a fraction of the earlier test which can be interpreted as a very successful implementation of the logarithmic transformation and the censored training data. On the other hand if you consider that the data is based on an unknown formula with no noise at all these results are still not satisfying. A closer look at the neural net and data is necessary to find further origins of the error.

To evaluate the performance of the neural net it is advisable to use the direct net outputs Prod.LogE and not the Prod.Cost values because the logarithmic transformation of errors could lead to mistaken interpretations. Imagine the error of target Prod.LogE = 7 would be +2 (see Table 14). After retransformation the absolute error is 7,006. Due to the logarithm the same error for a target Prod.LogE = 21 would lead to a retransformed error of 8,425,987,712. For the relative values it is vice versa. For increasing Prod.LogE an absolute error of +2 leads to a declining relative error but after logarithmic retransformation this relative error is always the same. Mathematical proof:

$$\log output_1 - \log target_1 = \log output_2 - \log target_2 = \log c$$

$$\log \frac{output_1}{target_1} = \log \frac{output_2}{target_2} = \log c$$

$$\frac{output_1}{target_1} = \frac{output_2}{target_2} = c$$

$$\frac{output_1}{target_1} - 1 = \frac{output_2}{target_2} - 1 = c - 1$$

$$\frac{output_1 - target_1}{target_1} = \frac{output_2 - target_2}{target_2} = c - 1$$

6.3.2.1 Rounding Errors

As with Test 1 some rounding errors might occur. Due to the logarithmic transformation they differ from those errors, which have been mentioned earlier:

- The logarithmic transformation of the production cost leads to numbers with up to 20 digits after the decimal point (maximum precision of Microsoft Excel spread-sheet program). When this data is copied into Qnet the numbers are rounded to 5 digits after the decimal point.

- Automatic normalisation and renormalisation of the data to train the network (see chapter 6.2.2)

- The results of the neural net have to be retransformed (Prod.Cost = $e^{Prod.LogE}$) via a spread-sheet program.

A comparison of the original production costs and the computed production costs leads to the result that the relative error never exceeds a value of more than 0.001%. Therefore a rounding error can be neglected.

Number	WT	MCPLXS	PSTART	Duration	QTY	Prod.Cost	Error
1167	1	4.518	34335	2922	1000	231574.13	
1178	1	4.518	34335	2922	1000	237468.69	2.55%
1234	1.5	4.326	34335	2922	1000	267504.33	
1252	1.5	4.326	34335	2922	1000	274212.98	2.51%
1302	1	4.823	34335	2922	1000	302341.61	
1321	1	4.823	34335	2922	1000	310215.77	2.60%
1335	1.5	4.518	34335	2922	1000	317476.14	
1345	1.5	4.518	34335	2922	1000	325550.23	2.54%
1375	1	4.963	34335	2922	1000	341822.78	
1386	1	4.963	34335	2922	1000	350822.01	2.63%
1457	2	4.518	34335	2922	1000	397128.82	
1468	2	4.518	34335	2922	1000	407219.06	2.54%
1551	2.5	4.518	34335	2922	1000	472436.42	
1566	2.5	4.518	34335	2922	1000	484428.09	2.54%
1561	1.5	4.984	34335	2922	1000	481337.13	
1585	1.5	4.984	34335	2922	1000	494020.01	2.63%
1604	1	4.518	34335	2922	3000	507265.49	
1623	1	4.518	34335	2922	3000	519442.94	2.40%
1656	3	4.518	34335	2922	1000	544454.06	
1671	3	4.518	34335	2922	1000	558259.40	2.54%

Table 15: Contradictory Data

6.3.2.2 Contradictory Data

A comparison of all data records with each other shows that the same input values for 326 samples lead to different outputs. Table 15 displays 20 examples where this contradiction occurs. The applied data set has been joined together out of different files which have been provided by Daimler-Benz Aerospace Airbus GmbH. One file is accidentally based on a different escalation file than the others. Escalation files contain tables with the expected inflation rate for the next couple of years in order to consider an increase of costs for long lasting projects that is based on the devaluation of money. PRICE Systems provides the user with escalation files that are set up for each country. Based on a different statistical analysis Airbus Industries uses its individual escalation file which is only slightly different to the one provided by PRICE. For this reason only long term projects (Duration = 2922) differ in their production costs.

Neural networks are generally very robust against scattering (see chapter 2.10) and therefore this error is hardly worth mentioning because the difference between the contradictory data records never exceeds 3%. A test run of a neural network with the same set up as described in this chapter but trained without contradictory

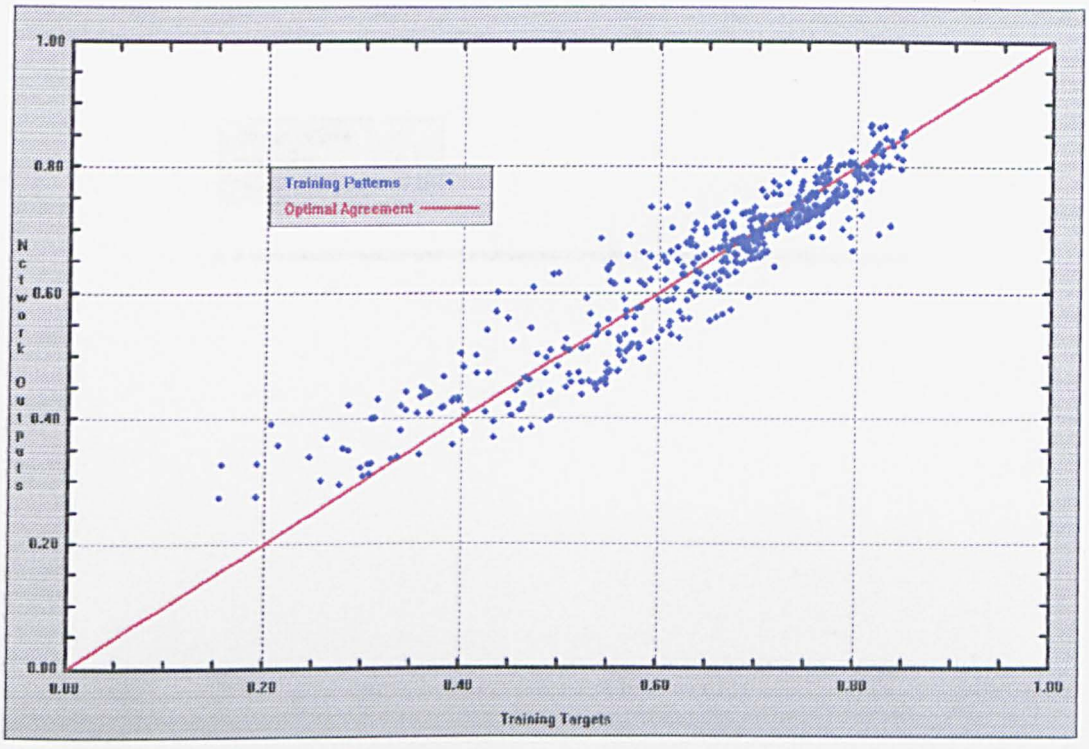


Figure 32:measured versus predicted Prod.CostE

data performed almost identically as the network which has been trained with contradictory data. These errors in the data therefore do not explain the bad performance of the network.

6.3.2.3 Distribution of Errors

Figure 32 displays training targets, which are the Prod.LogE values of the test data, and net outputs of Prod.LogE in a co-ordinate grid. The values on the x and y-axis lie in between 0 and 1 because the program automatically normalises the data into the range of these numbers. The optimal agreement is on the red line (training target = net output). It is very striking that those targets which are situated above the 70% mark are quite evenly spread around the optimal agreement and generally much closer to it than those underneath this mark. All values below the 30% mark are overestimated by the network.

In this example the network finds it easier to adapt to high outputs than low outputs. But in this case the reason for this cannot be the same as the one mentioned in chapter 6.2.2 (production costs are spread over several orders of magnitude) because the range of outputs is limited from 6 to 16. Rather the

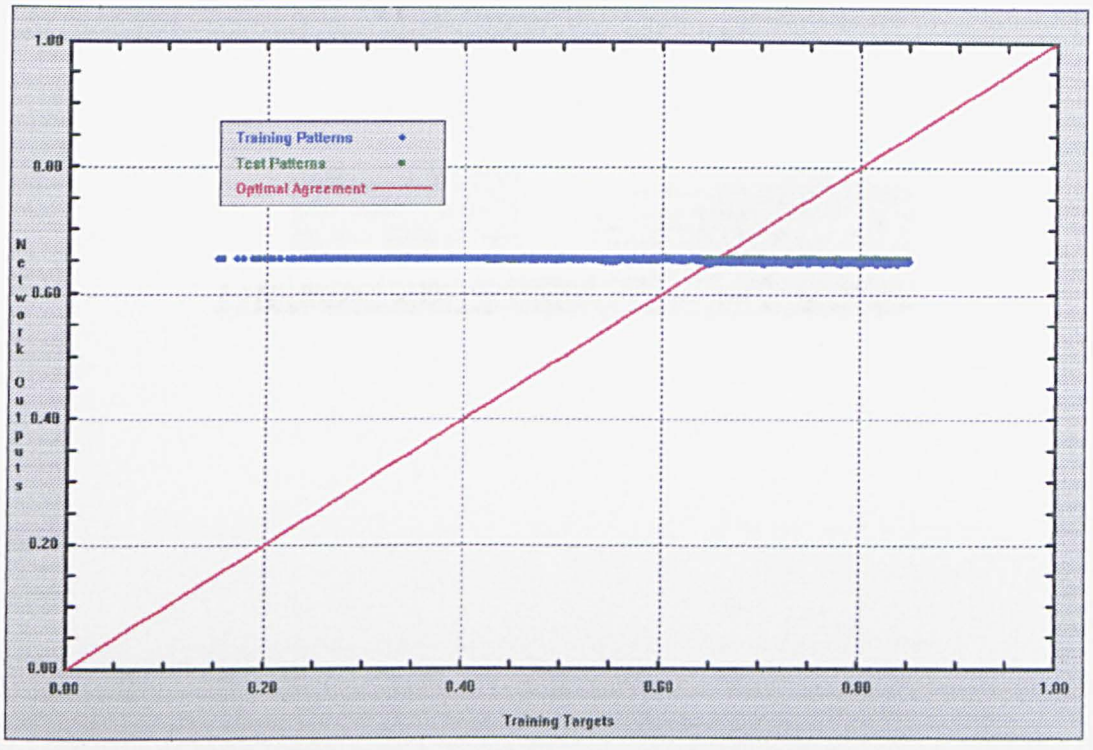


Figure 33: measured versus predicted Prod.CostE at beginning of training

observation of measured and predicted outputs in a co-ordinate grid during the training process can give an explanation of this phenomena.

In the first training step the neural net sets all outputs onto the same level and overestimates most of the patterns (see Figure 33). During the following learning steps the net mainly concentrates on those values that are to the right of the intersection of the training patterns line and the optimal agreement line. In this area the training patterns move toward the red line during the first 500 iterations (see Figure 34). While these outputs are further adapted the training patterns on the left side of the intersection begin to move down to the optimal agreement line. Small values on the very left have a long way to go in order to perform well.

Considering that an increasing amount of iterations results in a decreasing error of the network and this decrease leads to a smaller adjustment of the weights it is quite understandable that those outputs which have been adjusted at the beginning of the training adapt well and quickly. On the other hand those outputs which start their adaptation after about 1,000 iterations (i. e. small values) have hardly any chance to reach their optimal agreement because the learning rate then is just a fraction of initial rate. The fact that these values have a particularly long way to

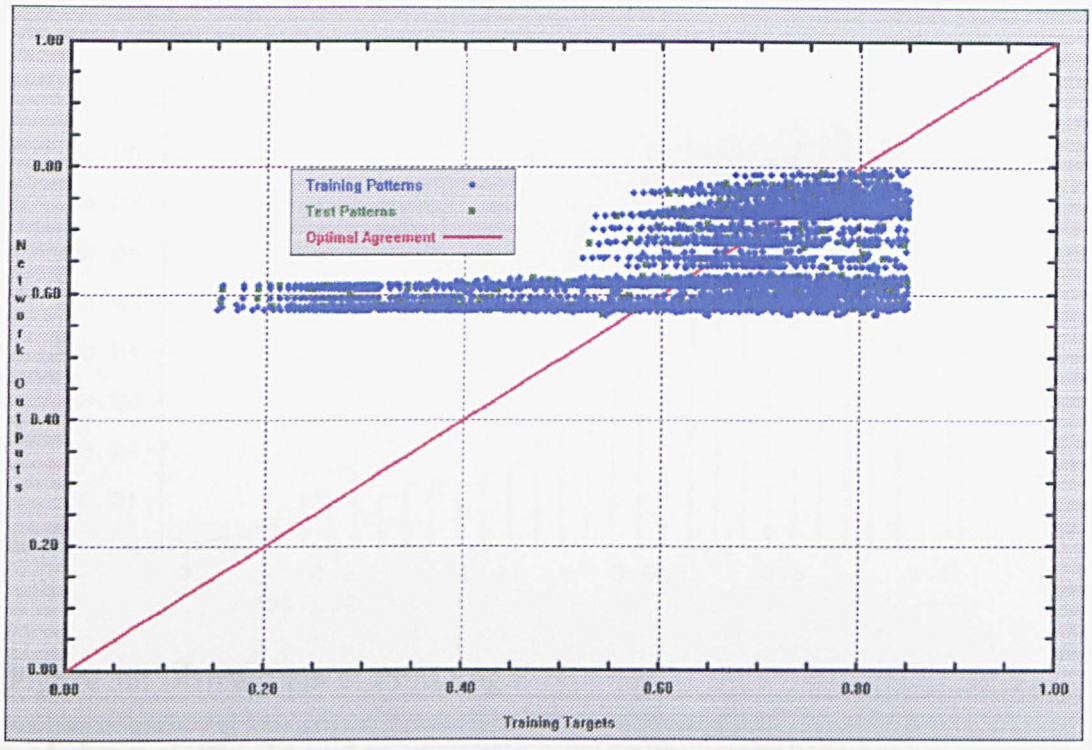


Figure 34: measured versus predicted Prod.CostE after 500 iterations

go makes the adaptation even more difficult and therefore even after about 100,000 iterations these outputs are still overestimated by the neural net.

The above paragraphs explain how the net distributes its outputs (and errors) during training but there is still a general contradiction in the learning behaviour. The gradient descending method implies that the neural net attempts to correct large errors prior to small errors (see chapter 6.2.2.2). This was the major reason why the net in chapter 6.2 performed so well for high outputs and very poorly for low values. This seems to be contradicted by the observation that smaller errors are corrected before larger errors.

If you apply this theoretical learning behaviour onto chapter 6.3 (Test 2) this would theoretically mean that within the first training iterations the net would have to adapt the low values first because they have the highest error in the net (normalised difference is about 0.6-0.2 = 0.4). As soon as these errors decrease the learning algorithm should include the adaptation of other (lower) errors. When reaching an average error of 0.2 the right side of the intersection of optimal agreement and training patterns should be simultaneously adapted. But in fact the net mainly concentrates on large outputs only at the beginning of the training as

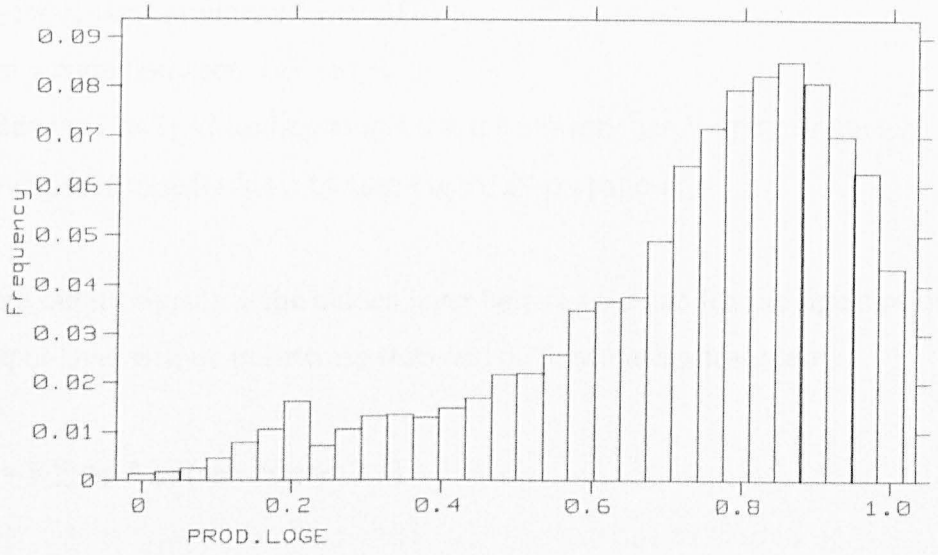


Figure 35: Distribution of Prod.Log.E

Figure 34 shows clearly. To find an explanation for contradictory behaviour a closer look at the data is required.

6.3.2.4 Distribution of data

An analysis of the output data shows that about 50% of all records are in the range of 0.7 to 0.9 with a mean output of around 0.6. Figure 35 displays as a histogram the output value Prod.Log.E of the test set after normalisation.

At the beginning of the training the network outputs are all almost equal (see Figure 33 on page 77). One of the reasons for this are the randomly chosen small weights of the network. Their values lie between -0.3 and +0.3. Even if the input parameters would all be 1 or 0 the output of each node in the hidden layer would have a value between 0.2 and 0.8. Mathematical proof:

$o_h = f(\sum w_i * i_i)$ (see chapter 2.3)

For $i = 0 \Rightarrow \sum w_i * i_i = 0 \Rightarrow f(0) = 0.5$

For $i = 1$ and $w = +0.3 \Rightarrow \sum w_i * i_i = +1.5 \Rightarrow f(+1.5) = 0.8$

For $i = 1$ and $w = -0.3 \Rightarrow \sum w_i * i_i = -1.5 \Rightarrow f(-1.5) = 0.2$

o_h = output of each neuron of hidden layer

i = input values (either all 1 or all 0)

w = weight (between -0.3 and +0.3)

index i = 1 to 5. According to this test the network has 5 input parameters.

f = sigmoid transfer function (see Figure 29 on page 77)

With output signals in the hidden layer between 0.2 and 0.8 the signal of the output layer will lie in between 0.25 and 0.75. Mathematical proof:

$o_o = f(\sum w_h * i_h)$ (see chapter 2.3)

For i = 0.8 and w = +0.3 $\Rightarrow \sum w_i * i_i = +1.2 \Rightarrow f(+1.2) = 0.75$

For i = 0.8 and w = -0.3 $\Rightarrow \sum w_i * i_i = -1.2 \Rightarrow f(-1.2) = 0.25$

The calculation for i = 0.2 or i = 0.5 is not carried out because these values would have a lower impact on the transfer function and their results would lie between 0.75 and 0.25.

o_o = output of neuron of output layer

i = output of hidden neuron (either all 0.2 or all 0.8)

w = weight (between -0.3 and +0.3)

index i = 1 to 5. According to this test the network has 5 hidden neurons.

f = sigmoid transfer function (see Figure 29 on page 77)

As a result you can say that if all input vectors would be either 0 or 1 the output would lie between 0.25 and 0.75. In fact the outputs lie within a much smaller ranger. There are two possible reasons for this:

1. All weights are chosen randomly and therefore they can take on either positive or negative values. When the output of the network is calculated these positive and negative values eliminate each other. In order to examine to which extent this random elimination can occur, evenly spread numbers have been generated for w and i and then the output of the network has been calculated according to

formula of the mathematical proof ($o = f(\sum w_i * i_i)$; $(-0.3 \leq w \leq +0.3; 0 \leq i \leq 1, {}_i = 5)$). This procedure has been repeated 10,000 times. The smallest value of these samples for output layer is 0.34575 and the biggest value is 0.66616. As a result the mutual elimination of positive and negative values shrinks the range down to 0.32041 (0.66616 - 0.34575).

2. The values of the weights are generated randomly only once during the initialisation process. When a new pattern is applied to the network these weights remain the same. Therefore the influence of the weights on each pattern is always the same. If you select randomly a set of weights for the network and then apply 10,000 patterns to the network according to the formula $o = f(\sum w_i * i_i)$ while the weights remain unchanged then the output of the hidden layer lies between 0.36862593 and 0.61585461 and the net-output lies between 0.59343997 and 0.60957969. This way the range of the outputs has a value of 0.01613972 (0.60957969 - 0.59343997).

In fact at the beginning of the training process all outputs of the network have almost the same value. Based on the gradient descending method during the training process a minimal adjustment of the weights leads to a high as possible reduction of the total error of the network. Because almost 50% of all outputs lie between 0.7 and 0.9 a relatively large part of the total error can be found on this interval. Even if each single error is rather small the gradient descending method adapts errors within the 0.7 to 0.9 interval first, because the correction of many small errors has a higher influence on the total error than the correction of a very few big errors as they occur for network outputs below 0.7. This explanation for the correction of errors does not contradict the explanation that is mentioned in section 6.2.2.2. Providing that the total error of a function is fairly high the gradient descending method attempts to correct high errors first. For this test on the other hand it can be assumed that the initial error of the neural net is not too high because almost 50% of all outputs lie between 0.7 and 0.9 and each single error within this range is relatively small. As soon as this total error becomes a little smaller it is not possible to recognise a general adaptation of the weights because particularly for non-linear neural networks the error function becomes too twisted and therefore the course of the gradient descending method cannot be

predicted. It is quite certain that (providing that the learning steps are sufficiently small) the total error is minimised until a (local) minimum has been reached.

If the outputs and errors of the network are distributed as mentioned in the above paragraph, the gradient descending method might not seem to be a suitable method to establish a neural net for cost estimation because relatively high errors can be disregarded during the learning phase. On the other hand a closer inspection of this learning behaviour may identify some advantages. The learning algorithm mainly concentrates onto those areas where many data samples are present (i. e. 0.7-9.9) and adapts these outputs particularly well. In the realm of cost estimation this can be very desirable because it can be assumed that future projects for which the costs are estimated will usually lie within the same cost range as those projects where many data samples already exist. The fine tuning of the learning algorithm within the cost range of the majority of all data samples can lead to fairly accurate estimates particularly for those projects which lie in the same cost range.

This characteristic is quite comparable with regression analysis to determine CER's. Like the neural network a regression analysis is also based on a number of real data samples and therefore its precision can only be verified and validated within this range. Although it is possible to interpolate the function of the regression analysis for any interval the validity of these new ranges of the cost estimation becomes quite questionable.

6.3.2.5 Normalisation

Qnet software program has a utility which can normalise the input data automatically. This automatic normalisation has been used for the last two tests. If this option is selected during training set-up, all data for the nodes in the input layer and/or training targets for the output layer will be normalised between the limits of 0.15 and 0.85[83]. An explicit reason for choosing 0.15 and 0.85 instead of

[83] Help Manual of Qnet ver. 2.1: Neural Network Training - Learning Modes, Vesta Services, Inc. 1994, 1995

the very common 0 and 1 as upper and lower limit could not be found in the user manual. A possible explanation could be, that Qnet offers the user the opportunity to gradually add new data after training. If the old data set has been normalised within the 0.15 and 0.85 borders the values of the new patterns can be up to 15% above or below the values of the old patterns without making a new normalisation and a new training of the data necessary because this new data could take on numbers between 0 and 0.15 or 0.85 and 1.

Generally it can be said that the narrower the interval within the data is normalised the more difficult it is for the neural net to find out the differences between the input data sets. For example, if all target data is between .01 and .02, it would be better to normalise the data over a wider range so that the network can better resolve and predict the targets. To give the learning algorithm the best chance to distinguish between the data sets a manual normalisation within the maximal range of 0 and 1 is advisable.

6.3.3 Conclusion

Neither rounding errors nor contradictory data can be the reason for the relative bad performance of the neural net. The examination of distribution of errors and distribution of data requires a closer insight into the field of backpropagation learning algorithm and normalisation and will be discussed in chapters 6.5, 6.6 and 6.7. The following test is confined to improving the normalisation of the data.

6.4 Test 3

6.4.1 Preparation

As discussed in chapter 6.3.2.5 in this test the data will be normalised manually in order to use the maximal normalisation range. According to the following formula input and output values are linearly normalised within the closed interval [0;1].

$$S_i = \frac{(X_i - \min X)}{(\max X - \min X)}$$

Notation:

X_i = value of the raw input variable X for the ith training case

S_i = standardised value corresponding to X

In chapter 2.9 some complex normalisation methods are mentioned. For this test the simplest technique will be applied on the available data (Rescaling).

Generally the data has to be prepared in such a manner that it becomes as easy as possible for the learning algorithm to recognise the differences between the training sets in order to establish the cost function. To make the distinction between the data sets as easy as possible in this test all values will be rescaled to the largest possible interval [0;1] instead of using the default normalisation of Qnet from 0.15 to 0.85.

In case the input values would be normally distributed and the standard deviation σ is relatively small it could be useful to allow those values which lie within the standard deviation σ (the majority of 68% of all data sets would be located there) a normalisation over a wider range (i. e. 0.2 to 0.8). Then the other values which are located outside σ have to be squeezed outside the range (i. e. 0-0.2 and 0.8-1). This kind of normalisation supports a fine tuning of relevant data (within the standard deviation) while extreme values and statistical outliers are hardly considered.

If the cost estimator has a good knowledge of those intervals which are particularly relevant for the cost function then he can speed up the learning algorithm by several orders of magnitude and reach a higher adaptation to the desired cost function by preparing the data appropriately.

As already mentioned in chapter 6 only a selection of possible input values has been made in order to limit the amount of data. For this selection the data has been chosen in such a manner that almost every combination of input values is

available. Numbers that are commonly used in practice have been particularly considered. Because the selection of the data is not random the data is not normally distributed. Therefore a rescaling of the data seems to be sufficient.

6.4.2 Results

This time the mean error of the predicted values of the test set is 58% (compared to 78% in test 2) and the absolute mean error is 772,975.72 (887,038.5 in test 2). The maximum error is 811% (1,400% in test 2). Again a considerable improvement of the neural net has been achieved but still this result is not satisfying.

6.4.3 Conclusion

Because the preparation of data can have such an impact on the learning behaviour of the neural net it might be advisable to examine the normalisation a little bit more carefully. Therefore the following chapter will give a more detailed insight into the normalisation of data.

6.5 Test 4

6.5.1 Preparation

As already mentioned in chapter 2.9 the purpose of normalisation is to create as optimal conditions as possible for the training of the neural net. Based on the input values the backpropagation algorithm calculates the output of the network, compares this output with the target value and then determines the error as difference between calculated and target value. Then the total error of the network is minimised with the gradient descending method.

Since the error of a neural network depends on the combination of its weights a functional relationship between weights and error exist. In order to explain the functioning of the gradient descending method graphically Figure 36 shows a

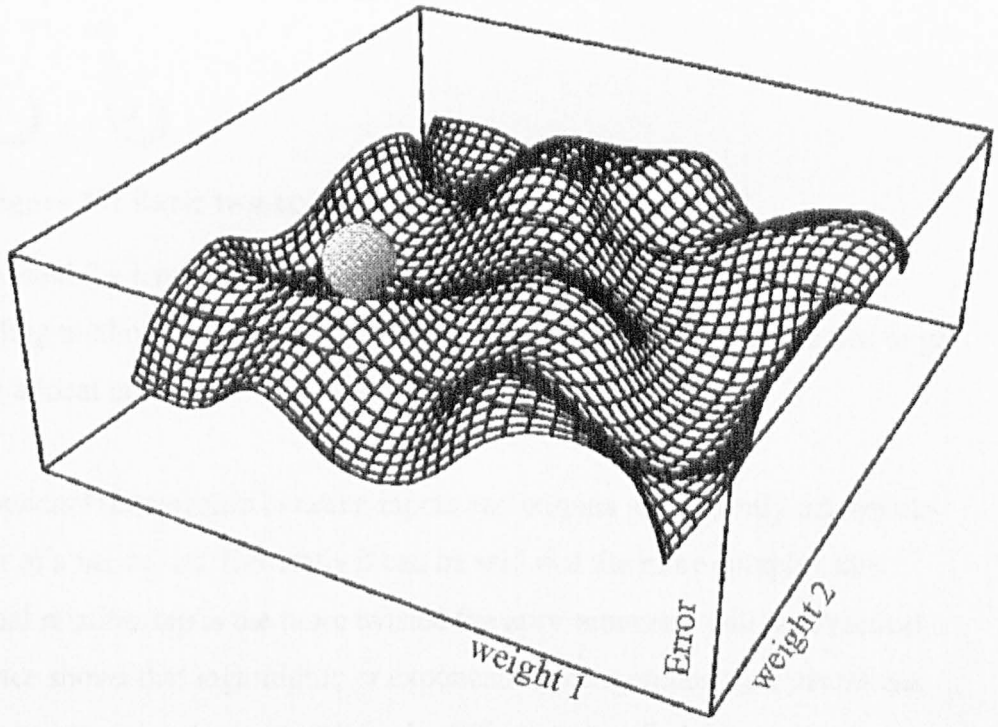


Figure 36: Three-dimensional error mountain

three-dimensional function with two weights as x- and y-values and the error of the network as z-value. Because this figure uses two weights only the error function is based on a very simple neural net, i. e. a two-layer network as it is displayed in Figure 37.

At the beginning of the training a random combination of weights is chosen which leads to a certain error. This initial phase can be symbolised with a ball that is randomly placed anywhere in the error mountain. The gradient descending method mathematically searches the direction where a minimal change of weights leads to maximal reduction of the error.

Graphically this procedure can be symbolised as a ball rolling down the steepest way down into a valley (see Figure 36). As soon as a local minimum has been found the learning phase is finished (the implementation of an alpha term is not considered in this example), because each minimal change of weights would lead to a higher error. Because a local minimum can still be very high (i. e. high error of the network) it is generally highly desirable to reach the absolute minimum (or a minimum that comes as close as possible to the absolute minimum). But the

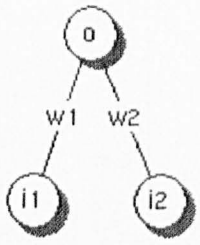


Figure 37: Basic two-layer network

more twisted the error mountain is the more difficult it is for the gradient descending method to find the absolute minimum and the higher is the risk to get stuck in a local minimum.

The functional relationship between inputs and outputs significantly determines the error of a neural net. Generally it can be said that the more complex this functional relationship is the more twisted the error mountain will be. Practical experience shows that logarithmic or exponential or trigonometric connections between inputs and outputs are relatively difficult to handle by neural networks. Simple relationships on the other hand (e.g. linear functions) lead to a relatively plain error mountain and therefore the backpropagation algorithm leads to a faster and better adaptation of the neural network.

If the cost estimator knows any functional relationships between inputs and outputs he can use this knowledge for the normalisation of the data in order to create as a simple error mountain as possible for the training phase.

If, for example, a square functional relationship between the weight of a component and the production cost exists, the data can be normalised with a square root function in order to create a linear relationship between normalised inputs (weight) and outputs (production cost).

Now the question can this knowledge help to create a better neural network in this test. To determine the relationship between each single variable and the output for each variable those data sets had been selected, where all other variables remain constant (except the one that is being investigated). Even though the author looked for as many data sets as possible for each group the yield was quite modest. For example only two different numbers for PSTART exist with all other

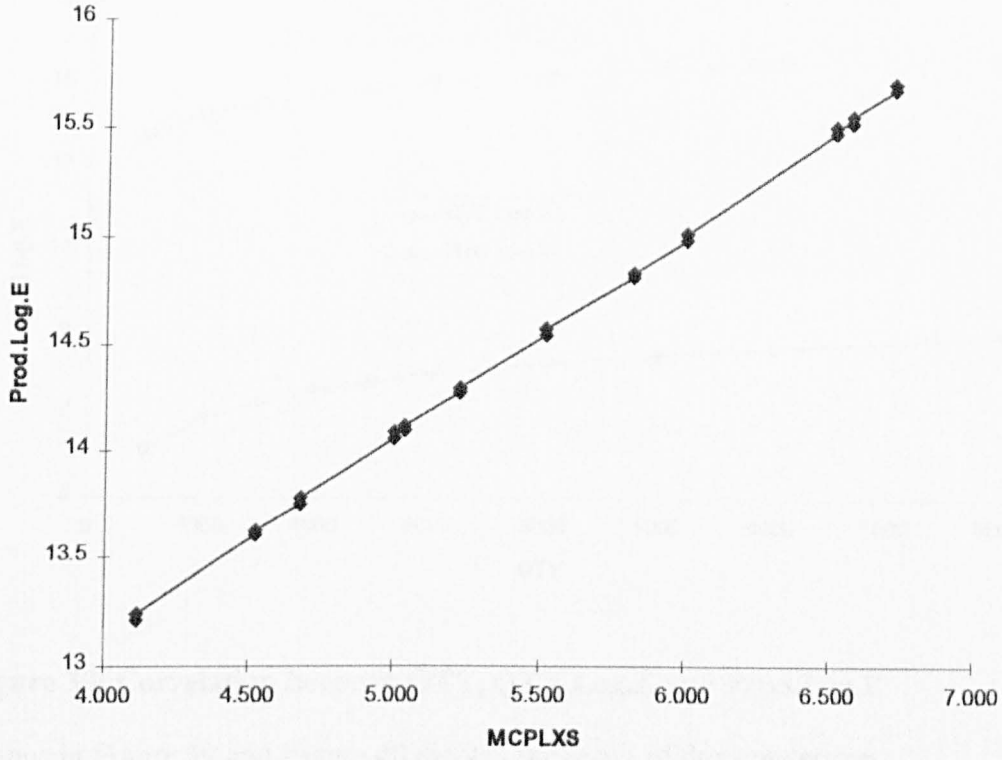


Figure 38: Correlation between MCPLXS and Prod.Log.E

values (MCPLXS, WT, Duration and QTY) remain constant. Because two numbers are not sufficient to determine a functional relationship no correlation analysis was carried out for this variable.

Figure 38 shows the dependence between MCPLXS and Prod.Log.E. The straight line can be interpreted as linear connection between these variables. The mathematical correlation of MCPLXS and Prod.Log.E is 99.982%. This linear function can be regarded as optimal configuration for the network training and therefore a further adaptation is not necessary.

Figure 39 and Figure 40 display the influence of QTY and WT respectively on Prod.Log.E (lines with squares). In these examples a logarithmic or a root function could describe the connection between input and output. The statistical analysis of the production cost (not Prod.Log.E) with QTY and WT resulted in a linear correlation of 99.953% for QTY/Production Cost and 99.992% for WT/Production Cost. In order to get a linear correlation towards Prod.Log.E the variables (WT and QTY) simply have to be converted with $\log_e$. The lines with

Figure 39: Correlation between QTY, QTY.Log.E and Prod.Log.E

the rhombus in Figure 39 and Figure 40 display the result of this conversion (QTY.Log.E and WT.Log.E). The parallel course of QTY.Log.E and Prod.Log.E, and WT.Log.E and Prod.Log.E indicates the linear correlation.

Duration can be adapted in a similar manner to QTY and WT. However with the $\log_{10}$ a much higher linear correlation can (93.697%) be reached than with $\log_e$. Figure 41 displays the parallel course of Dur.Log.10 and Prod.Log.E. The disadvantage of this variable is that the correlation analysis is based on 5 values only. Therefore it is not sure if this correlation can be transferred onto the whole data set.

The objective of this test is to reach as linear a relationship between inputs and output as possible. Therefore WT, QTY and Prod.Cost will be normalised with Log_e, duration will be normalised with $\log_{10}$ and MCPLXS and PSTART will not be changed. After this step all values are rescaled to an interval from 0 to 1 according to the formula in chapter 6.4.1

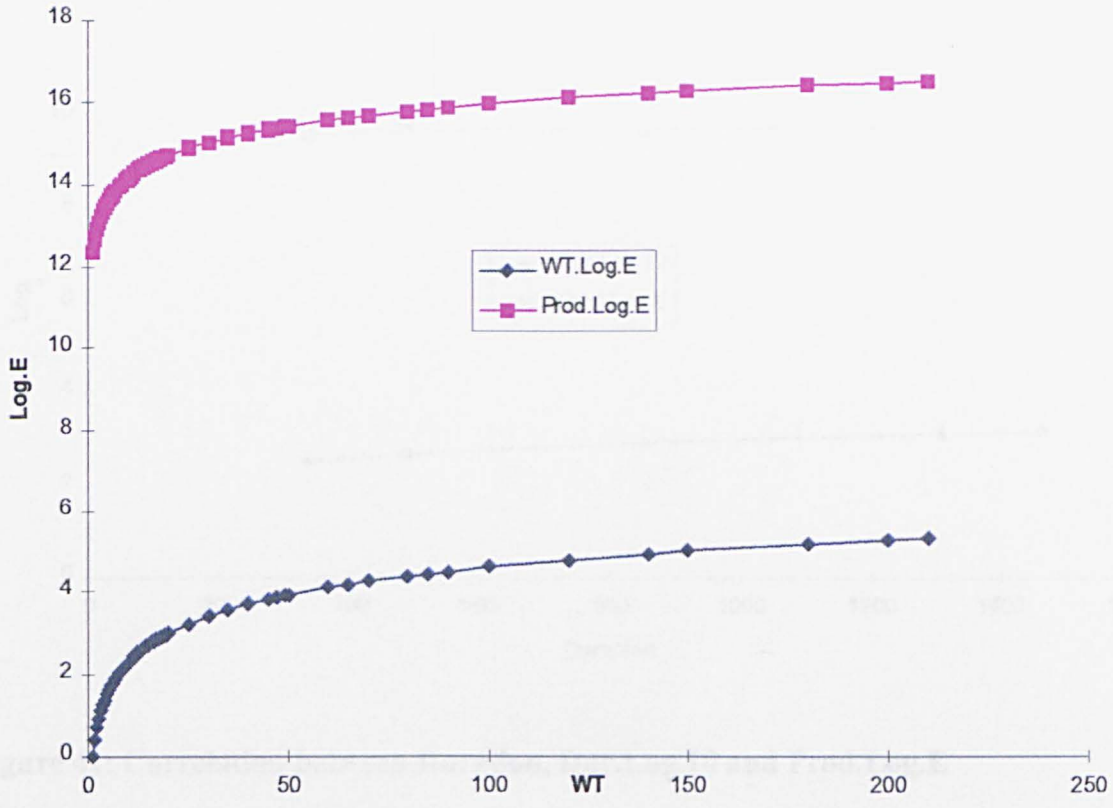


Figure 40: Correlation between WT, WT.Log.E and Prod.Log.E

6.5.2 Results

These results surpass the other tests by far. The mean error of the predicted values of the test set is only 19.20% (compared to 58% in test 3) and the absolute mean error is 533,358.66 (772,975.72 in test 3). The maximum error is 127.11% (811% in test 3).

6.5.3 Conclusion

This relatively good result does not give any reason for great satisfaction. In real world applications the quality requirements for cost estimates is so high that an average error of about 20% in the aeroplane industry is just acceptable (see chapter 3.5.7) Considering that the data material is based on a mathematical equation with no scattering at all it can be assumed that it may be much more problematical for a neural net to determine a function out of real world data where scattering is most likely included. The existence of a mathematical function

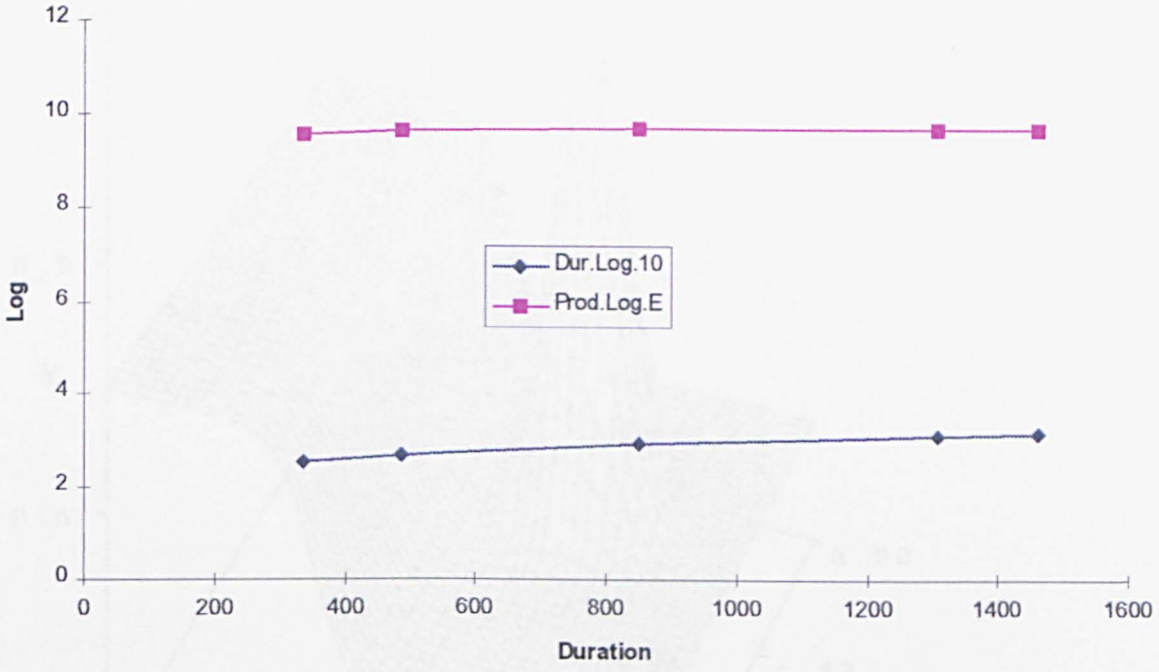


Figure 41: Correlation between Duration, Dur.Log.10 and Prod.Log.E

helped in this test to increase the quality of the results and strictly the applied normalisation of chapter 6.5.1 is not permissible for the following reason:

In order to determine the correlation between each variable and the output for each variable those data sets have been selected, for which all other variables remained constant. In spite of the attempt to find as many data sets as possible in order to reach a good adaptation for each variable the number of applicable samples was quite small and it is only up to a limited extend possible to verify whether the correlation analysis keeps the same validity over the complete definition area.

Input	**Data for correlation analysis**		**Complete data**	
	Min	**Max**	**Min**	**Max**
MCPLXS	4.112	6.718	3.518	8.069
QTY	500	8000	1	8000
Duration	335	1461	335	3287
WT	1	210	1	1900

Table 16: Data range of correlation analysis and total data

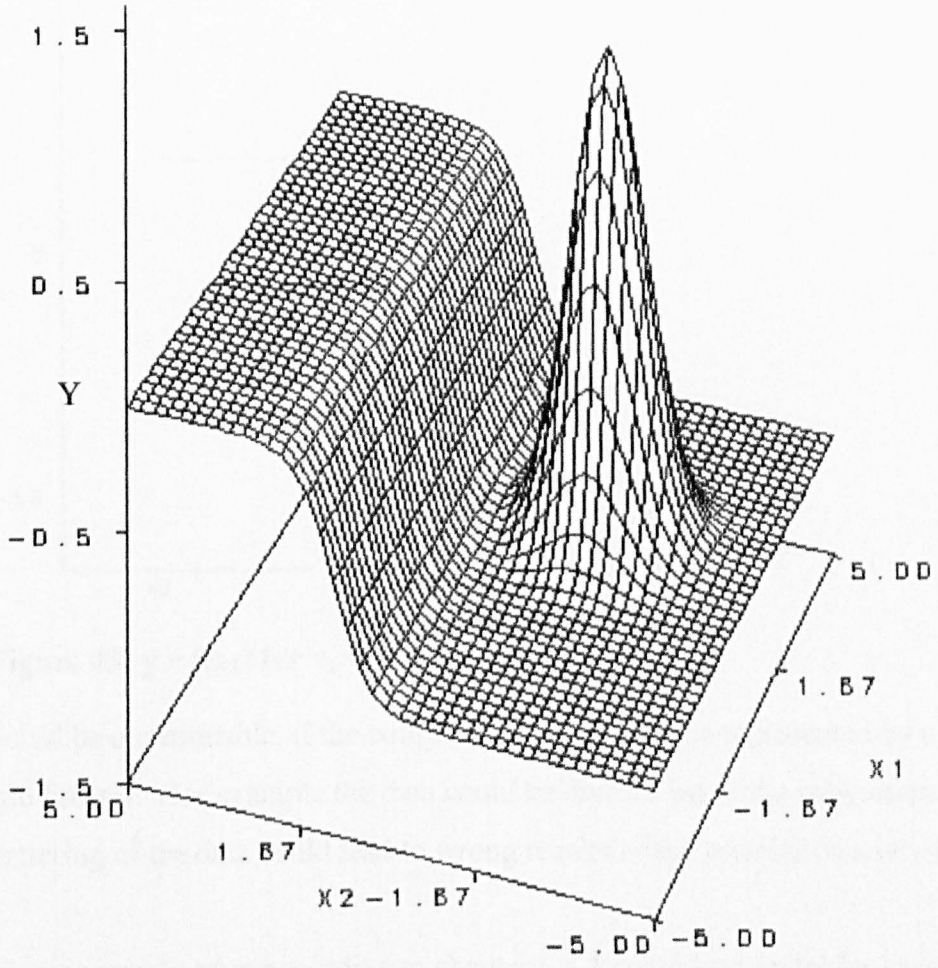


Figure 42: Three-dimensional example

Table 16 shows the complete definition area of each variable and the interval on which the correlation analysis has been carried out. For example the interval of the correlation analysis for MCPLXS is from 4.112 to 6.718 while the whole data set has complexities from 3.518 to 8.069. Those areas which have been excluded from the correlation analysis (from 3.518 to 4.112 and from 6.718 to 8.069) could theoretically have some other kind of correlation or no correlation at all. Therefore it is statistically incorrect to extend the conclusions from an analysis of part of the data to the complete data.

On the other hand the results that have been achieved with this procedure are good. The major reason for this is that the present data material is based on a mathematical function and therefore a partial correlation analysis is very likely to be transferable to the whole range. Assuming the analysis was based on a real example then

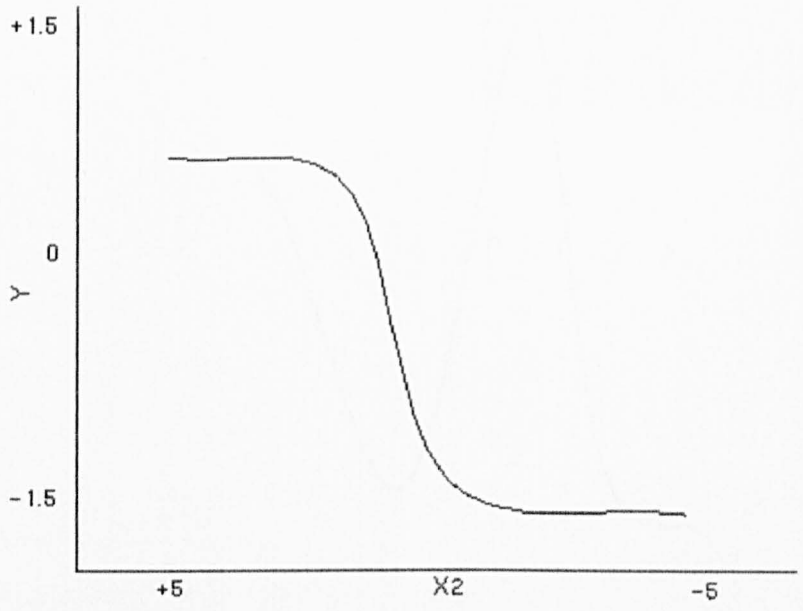


Figure 43: $y = f(x_2)$ for $x_1 = -1.67$

- it would be questionable, if the complete data set could be represented by a single function (for example the data could be divided into little subsections).
- a scattering of the data could lead to wrong results of the correlation analysis.

Therefore a normalisation according to chapter 6.5.1 could lead to dubious results if real world data is applied. But there is another reason why this procedure might not be applicable.

The premise of the correlation analysis between one input and output is that the other inputs remain constant. But depending on the chosen values of the constant inputs the correlation between the variable input and output can drastically change.

Figure 42 shows a function with X1 and X2 as input variables and output Y. For example, to determine the correlation between X2 and Y. X1 is set constant at -1.67. The functional relationship between X2 and Y corresponds to a cut through X1 = -1.67 and is shown in Figure 43. Assuming that X1 was set to +1.67 then this cut would have a completely different function for X2 and Y as it is shown in Figure 44.

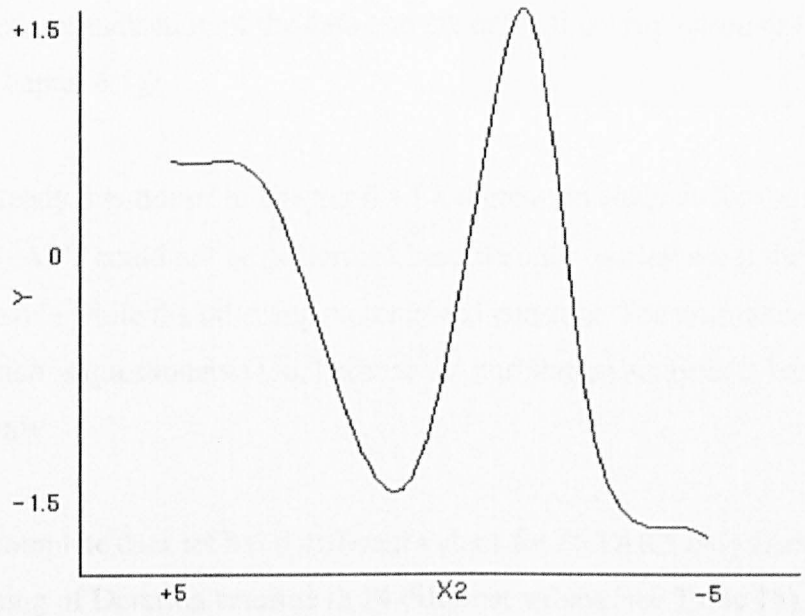


Figure 44: $y = f(x_2)$ for $x_1 = +1.67$

Theoretically it would be possible to check if the functional relationship between input and output for different combinations of constant inputs remains the same. However this requires extensive data material which is usually not accessible in real life applications. If this check leads to different correlations then a normalisation according to this chapter must not be performed because no strictly determinable correlation exists.

Based on the present data the functional relationship of different combinations of constant inputs always remains the same. This is not really surprising since the data is based on a mathematical function and it can be assumed that the calculated relationships keep their validity for the complete interval. For real world data such a behaviour is very unlikely.

6.6 Test 5

6.6.1 Preparation

The aim of this test is not to improve the results of chapter 6.5. Rather another origin of the error of the neural net should be investigated. The question is; why can a neural network still have an average error of 20% with possibly the best

applied normalisation of the data and the optimal configuration of the network (see chapter 6.1)?

As already mentioned in chapter 6.4.1 a regression analysis for the normalisation of PSTART could not be performed because only two values at the most were accessible while the other inputs remained constant. The normalisation of the Duration is questionable too, because the correlation analysis is based on 5 data sets only.

The complete data set has 4 different values for PSTART only (see Table 17). A grouping of Duration resulted in 14 different values (see Table 18). Assuming that the data is evenly scattered up to 56 (14*4) combinations of PSTART and Duration should exist. The examination of the data on the other hand lead to the result that only 15 combinations of PSTART and Duration are present. This means that only one PSTART is assigned to each Duration (see Table 19). An exception is a Duration of 1461 which is associated with two values of PSTART.

Even though these variables are linearly independent and many combinations could be possible the neural network will assume a strong dependency between PSTART and Duration because the available data presents only a very limited number of combination.

The elimination of this "fake" dependency can lead to much better results as the following three examples will show. In Table 19 those three groups which have the highest amount of data sets have been selected (highlighted in bold type) and for each of these groups an individual neural network with QTY, MCPLXS and WT as input parameters and Prod.Cost as output parameter has been trained. The normalisation method is based on chapter 6.5.1. For testing reasons 200 data sets have been randomly excluded from each network.

PSTART	Amount
33970	325
34335	1635
34700	99
35065	2939

Table 17: Amount of different PSTART in total data sets

Duration	Amount
335	1249
486	63
700	315
730	142
821	46
851	63
1065	46
1308	63
1461	1136
2191	140
2192	99
2557	99
2922	1212
3287	325

Table 18: Amount of different Durations in total data sets

PSTART	Duration	No. of data sets	
35065	**335**	**1249**	**Test 5.3**
35065	486	63	
35065	700	315	
34335	730	142	
35065	821	46	
35065	851	63	
35065	1065	46	
35065	1308	63	
34335	1461	141	
35065	**1461**	**995**	**Test 5.1**
34335	2191	140	
35065	2192	99	
34700	2557	99	
34335	**2922**	**1212**	**Test 5.2**
33970	3287	325	

Table 19: Combinations of PSTART and Duration in total data sets

6.6.2 Results

The quality of the results of the networks is comparable to the network in chapter 5.2 which has been used to calculate the complexity. Table 20, Table 21 and Table 22 show a relative mean error of 2.39%, 5.69% and 4.61%. The relative maximum error never exceeds 33.2%. Test 5.2 has got the worst results. One reason for this could be the following:

According to chapter 6.3.2.2 there are 326 contradictory data sets in the data. They are all present in Test 5.2, because their PSTART equals 34335 and their Duration equals 2922. Their average deviation is 2.5%. Since this contradictory data has the effect of scattering, the function will probably find its own level in between the ordinary data space and the contradictory data space. Therefore it can be assumed that the elimination of the contradictory data will reduce the error so that the networks performance comes close the performance of Test 5.3.

It is quite striking that the relative mean error of test 5.1 is about 2% lower than those of test 5.2 and test 5.3. Since the other values (relative maximum error and absolute maximum error) are located at the same level a further analysis of this phenomena is not necessary.

6.6.3 Conclusion

The tests in this chapter show clearly that PSTART and Duration have a rather disturbing influence on the network's performance and do not give any contribution to finding the cost function. The reason for this is that (with one exception) for every Duration only one particular PSTART is assigned to. Therefore it is impossible for the neural network to determine a functional relationship between PSTART and Prod.Cost because for each PSTART the Duration has a different influence onto the production costs.

	absolute	relative
mean Error	50,078.25	2.39%
relative max. Error	163.93	22.37%
absolute max. Error	3,086,633.91	21.04%

Table 20: Results of Testdata 5.1

	absolute	relative
mean Error	297,879.82	5.69%
relative max. Error	79,200.21	33.20%
absolute max. Error	3,038,601.83	21.60%

Table 21: Results of Testdata 5.2

	absolute	relative
mean Error	101,949.70	4.61%
relative max. Error	314.45	27.87%
absolute max. Error	3,629,565.57	25.56%

Table 22: Results of Testdata 5.3

With the elimination of PSTART and Duration the networks quality became comparable to those in chapter 5.2. If there are still minor deviations to the results in chapter 5.2 the reason for this could be found in much more complex functional relationships for the determination of Prod.Cost than the determination of the complexity in chapter 5.2.

6.7 Test 6

6.7.1 Preparation

It would have been difficult to proceed with a normalisation according to chapter 6.6 with real world data. The reasons for that can be found in chapter 6.5.3 and therefore it can be said that the good results have been achieved by "cheating".

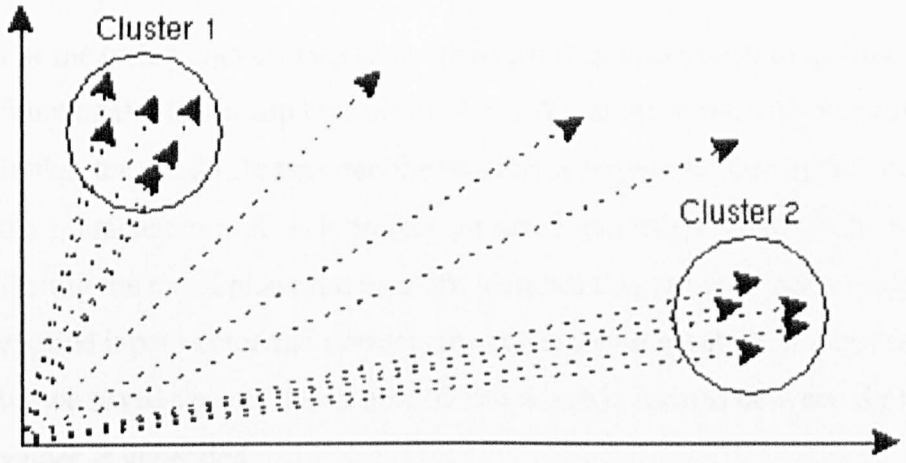


Figure 45: vectors with clusters

Still it should be possible to find a general normalisation method which leads to a satisfying quality of the results. To understand the following alternative another brief outlook on neural networks is necessary.

Within the last chapters it was mentioned that a neural network is trained until it represents a determined function. This does not mean that the function is saved in form of an equation in the network. Rather each combination of inputs produces a particular output. The sum of all inputs and outputs represents a function.

Assuming the training data consists of numerical values which contain considerably big gaps between the data sets then the network develops empty spaces between the training data. If during the recall phase an input value is chosen which lies exactly between two distant training values then the output will be ideally located between the outputs of the training data. As already mentioned in chapter 4.2 a neural network is capable of representing any hyper space (while a linear correlation analysis represents a hyper plane). Strictly speaking the network saves the trained data. If a new input vector is applied the network is able to calculate a new output and since to each input vector one output vector is applied the impression of a steady function comes into being. In fact just the information of some vectors is saved in the synapses which represents the hyper room. The network can use this information to calculate any vectors within the empty spaces. Any of these vectors or a group of vectors within narrow bounds are called a cluster (see Figure 45).

For the following normalisation method it is not necessary to assume any functional relationship between inputs and outputs in order to determine the costs. Rather the similarity between the projects is important. During the training phase the characteristics of each project are saved as a single vector in the network. During the recall phase the network searches that vector which mostly fits the applied input vector and presents the corresponding output. If a test set is similar to two saved vectors then a new output which is located between the other two values is generated.

If a neural network is looked at from this point of view, a normalisation method should be chosen which spreads the training values as evenly as possible over the complete hyper room to make a distinction between different values as easy as possible. Therefore the data is normalised according to the following rule:

1. Determine the maximum and minimum value of a variable
2. Determine the number of different values for this variable
3. The minimum value is set to 0 and the maximum value set to 1
4. The normalised distance between each group of values is calculated as: normalised distance = 1/(amount of different values - 1)

Example:

QTY can take on 19 different values (see Table 23). QTY = 1 is mapped to 0 and QTY = 8000 is mapped to 1. The normalised distance is 0.05555555555556 (=1/(19-1); see column 3 in Table 23). The remaining input variables are normalised. For example, when the QTY goes 80, the sixth value in the column, the normalised value is (6-1)* 0.05555555555556 which is equal to 0.2777777777778.

For the outputs the normalisation method of chapter 6.4 is applied. If the outputs were normalised as the inputs in this chapter it would be very awkward to renormalise the predicted values of the network because they usually do not hit exactly the measured values. To renormalise a measured value that lies between

QTY	Amount	normalised
1	503	0
5	126	0.05555555555556
10	191	0.1111111111111
20	189	0.1666666666667
50	311	0.2222222222222
80	124	0.2777777777778
100	362	0.3333333333333
200	173	0.3888888888889
500	882	0.4444444444444
800	440	0.5
1000	626	0.5555555555556
1500	123	0.6111111111111
2000	208	0.6666666666667
2500	119	0.7222222222222
3000	197	0.7777777777778
4000	150	0.8333333333333
5000	74	0.8888888888889
6000	70	0.9444444444444
8000	130	1

Table 23: Amount of different QTY in total data

two measured values a linear interpolation between these numbers would be required. This very time consuming procedure would hardly improve the quality of the predictions and therefore the normalisation method of chapter 6.4 is applied.

6.7.2 Results

The quality of the results of this test is quite comparable to chapter 6.6. The mean error of the predicted values of the test set is 22.21% (compared to 19.20% in test 4) and the absolute mean error is 541,303.48 (533,358.66 in test 4). The maximum error is 297.58% (127.11% in test 4).

6.7.3 Conclusion

Excepting the results of test 5 which were only achievable because it was known from the start that the data is based on a clearly defined mathematical function it can be asserted that test 6 achieved the best results of the six tests described in this chapter.

Because this normalisation method evenly spreads the data over the hyper space those clusters where many vectors lie closely together will be stretched while other spaces where only a few vector are present will be compressed.

For real world data it can be assumed that more data sets are available for those projects which lie within the typical cost range whereas for particularly big or small project hardly any data exists. The new normalisation method can have a favourable effect on both these data sets. For those ranges where many finely tuned data samples exist (typical projects) this normalisation method trains the network particularly well. At the edge of these ranges where only rough information is available this normalisation graduates the data only crudely.

This simple normalisation method neglects any functional relationships between inputs and outputs. Only the similarity is decisive for the cost estimate. This general procedure makes it easy to transfer the normalisation method onto other data sets. It allows the neural network to compare the characteristics between the projects because the relative values of the characteristics of the project can be modified.

6.8 Discussion of the test results

The data used in these tests make it much more difficult to generate a neural network which is able to determine a correct connection between input and output. Because Qnet has much more powerful tools to analyse the neural net this software program has been used instead of NNModel. In some tests on the other hand NNModel was also applied to ensure that the results of both software programs were comparable.

Although the dimensioning of the neural net was relatively easy, the main problem was the preparation (normalisation) of the data.

First of all the values of Prod.Cost covered several orders of magnitude (see chapter 6.2.2.1) and significant rounding errors occurred. Next to that some data

sets made the training much more complicated (see chapter 6.2.2.2) and therefore they had been eliminated out of the training set.

For the backpropagation algorithm it turned out to be very difficult to handle overlapping exponential or logarithmic correlations. The quality of the network could be considerably improved where these correlations were linearised (see chapter 6.5).

In chapter 6.2.1 it had been recognised that the combination of PSTART and PEND does not provide any useful information for the neural network and therefore PEND had been transformed into Duration. The advantage is that a mathematical correlation between Duration (or PSTART) and Prod.Cost exists. The disadvantage is that these values rather confuse the network than provide any useful information (see chapter 6.6). For this reason a comparable quality to the determination of MCPLXS could not be reached. This problem is practical for cost estimators who need to

1. find the main cost drivers
2. determine a functional relationship between cost drivers and costs
3. collect sufficient data in order to verify the functional relationships.

It is not only the above mentioned normalisation methods which have been applied to the network but also other procedures. Most of them are mentioned in chapter 2.9 (Standardisation of Column vectors, Standardisation of Row vectors, non-linear transformation). Their effects however did not improve the networks' performances.

All tests in this chapter are based on artificially generated data. Therefore the good results are not really surprising. But how would a neural network perform if real world data is used and if it is not certain whether any functional relationships exist at all? Particularly the normalisation method of chapter 6.7 has to prove if its universal usability is still valid.

7 Multi-Layer Perceptron on real world data

In chapter 5 and chapter 6 the Multi-Layer-Perceptron has been used on data which was generated by the PRICE model. Therefore the neural network has emulated an already existing mathematical model. But how would a Multi-Layer-Perceptron perform if it was trained with genuine data? Is the neural network still capable of recognising an existing connection between inputs and outputs? If the answer is yes, is the structure of this relationship good enough to perform a sufficiently precise cost estimate? To answer these questions in this chapter the network will be applied on real world data from Daimler-Benz Aerospace Airbus (DASA).

7.1 The Enterprise Daimler-Benz Aerospace Airbus

Daimler-Benz Aerospace Airbus GmbH is the German partner in international

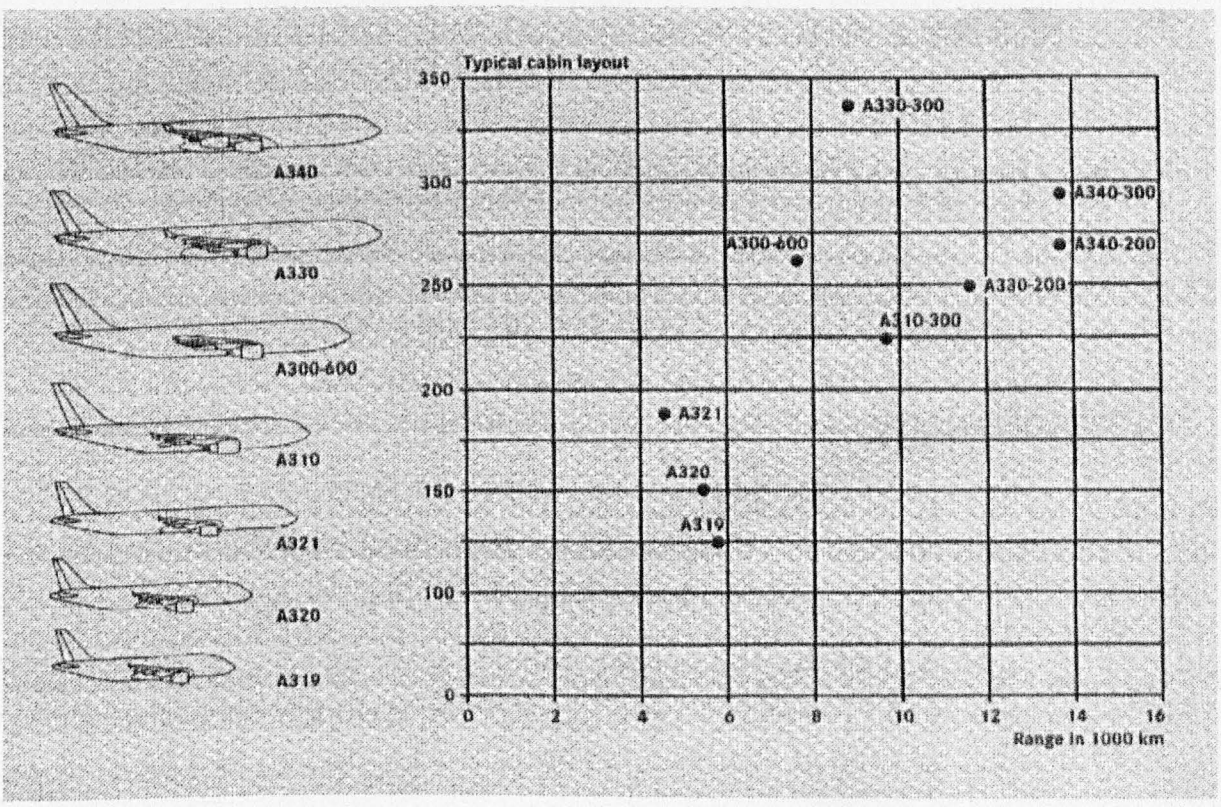


Figure 46: Different models of the "Airbus Family"

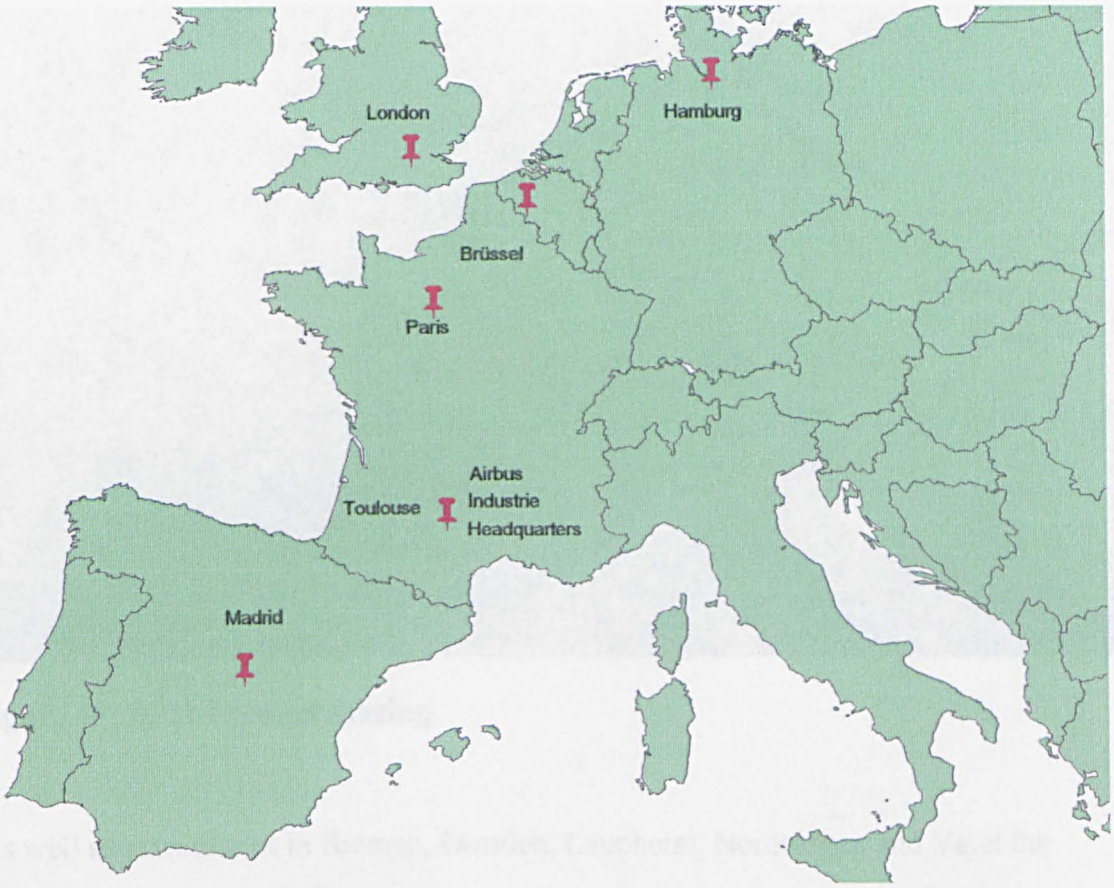


Figure 47: Airbus Industries Partners

civil and wide-body aircraft programs. As one of two major partners in the European Airbus program, DASA has been responsible from the beginning for the development and manufacture of major assemblies for all Airbus types.

Today "Airbus" stands for a family of aircraft accommodating from 120 to 350 passengers, covering short hauls and long ranges. Figure 46 shows the different models of the "Airbus Family" and classifies each type by the range and the cabin layout.[84]

With its four partners - Aerospatiale in France (37.9%), Daimler-Benz Aerospace Airbus in Germany (37.9%), British Aerospace in Great Britain (20.0%) and CASA in Spain (4.2%) - and two associate members Airbus Industries is one of the largest industrial undertakings in Europe. Each partner is fully responsible for its own components (see Figure 47).[85]

---

[84] Daimler-Benz Aerospace Airbus - prospect: Into the future, Hamburg, Nov. 1996, p. 2

[85] Daimler-Benz Aerospace Airbus - prospect: Into the future, Hamburg, Nov. 1996, p. 4

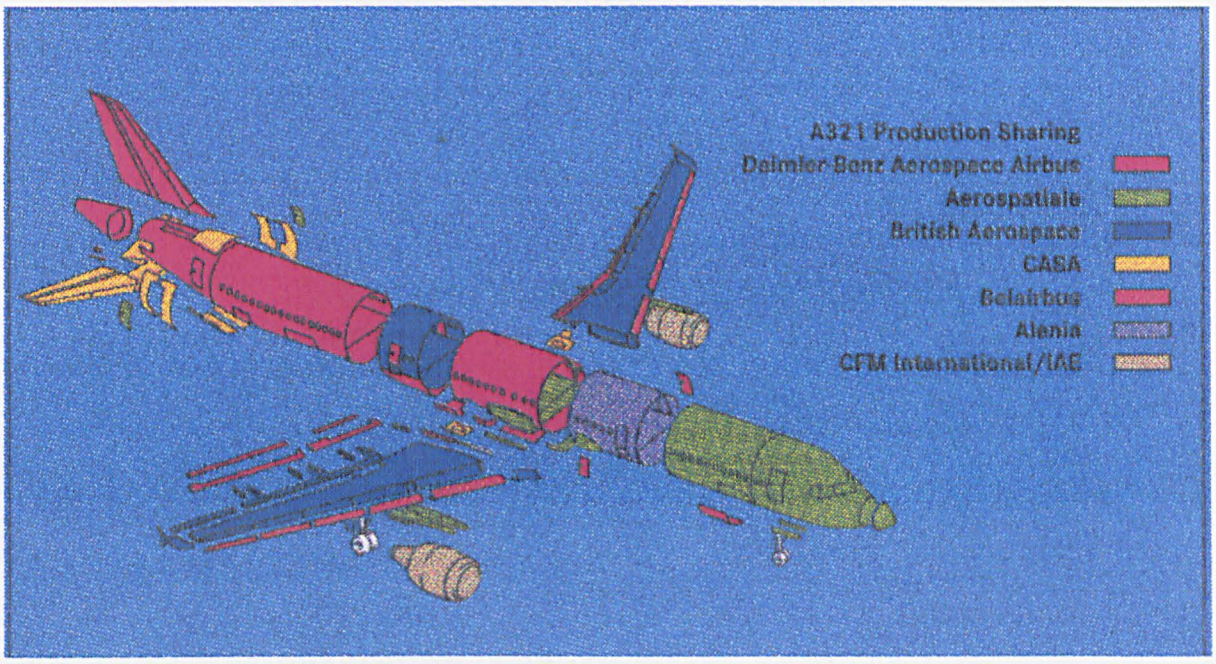


Figure 48: A321 Product sharing

As well as subsidiaries in Bremen, Dresden, Laupheim, Nordenham and Varel the Hamburg plant is the DASA management headquarters. The Hamburg plant is one of two assembly centres, the other being in Toulouse. In Hamburg the A319 and A321 single aisle aircraft undergo final assembly and are then delivered to customers. Hamburg is also home of the Development Department, Design Office, Product Managers of the various Airbus programs and Customer Service.

It is at the Hamburg plant that the fuselage sections for all Airbus aircraft are assembled and equipped with all vital systems such as electrical power, electronics, hydraulics, air conditioning and water. The company also has extensive test facilities including installations for static and dynamic testing.[86] Figure 48 displays the product sharing of the A321.

7.2 Cost Estimating at Daimler-Benz Aerospace Airbus

To achieve the most accurate cost estimates the aeroplane is divided into work packages. Table 24 displays all work packages for the A320. Each of these work packages is further subdivided into smaller work packages. This procedure is

Work-package No.	Work-package
1	Forward nose fuselage
2	Forward fuselage
3	Center fuselage
4	Aft center fuselage
5	Rear fuselage
6	Tailcone forward section
7	Tailcone rear section
8	Torque box
9	Wing box
10	Control surfaces
11	Fin and rudder
12	Tailplane and elevators
13	Pylons
14	Nacelles
15	Fan reversers
16	Landing gear
17	Furnishings
18	Main assembly
19	Join up forward fuselage (sections 11/12 + 13/14)
20	Join up forward fuselage / torque box (sections 15 + 21)
21	Join up rear fuselage (sections 18 + 19)
22	Join up center / rear fuselage (sections 15/21 + 16/17 + 18/19)
23	Power package
24	Wing assembly
25	Harness forward
26	Harness aft

Table 24: Work packages for A 320

repeated until an elemental level has been reached and another break-down would not provide any more useful information. Figure 49 shows a forward fuselage (work package no. 2 of the A320 in Table 24) which is further divided into the subsections B1, B2 and B3.

For many of these elements a parametric cost estimate with PRICE is performed. Depending on the kind of component other estimating methods can be applied as well. In addition DASA has an extensive real cost database at its disposal and therefore for many components these costs can be applied instead of performing a cost estimate. To calculate the complete costs of the complete aeroplane all prices of all elements are aggregated (considering the assembly outlay) to the top level.

[86] Daimler-Benz Aerospace Airbus information prospect, Hamburg, Nov. 1997, p. 12

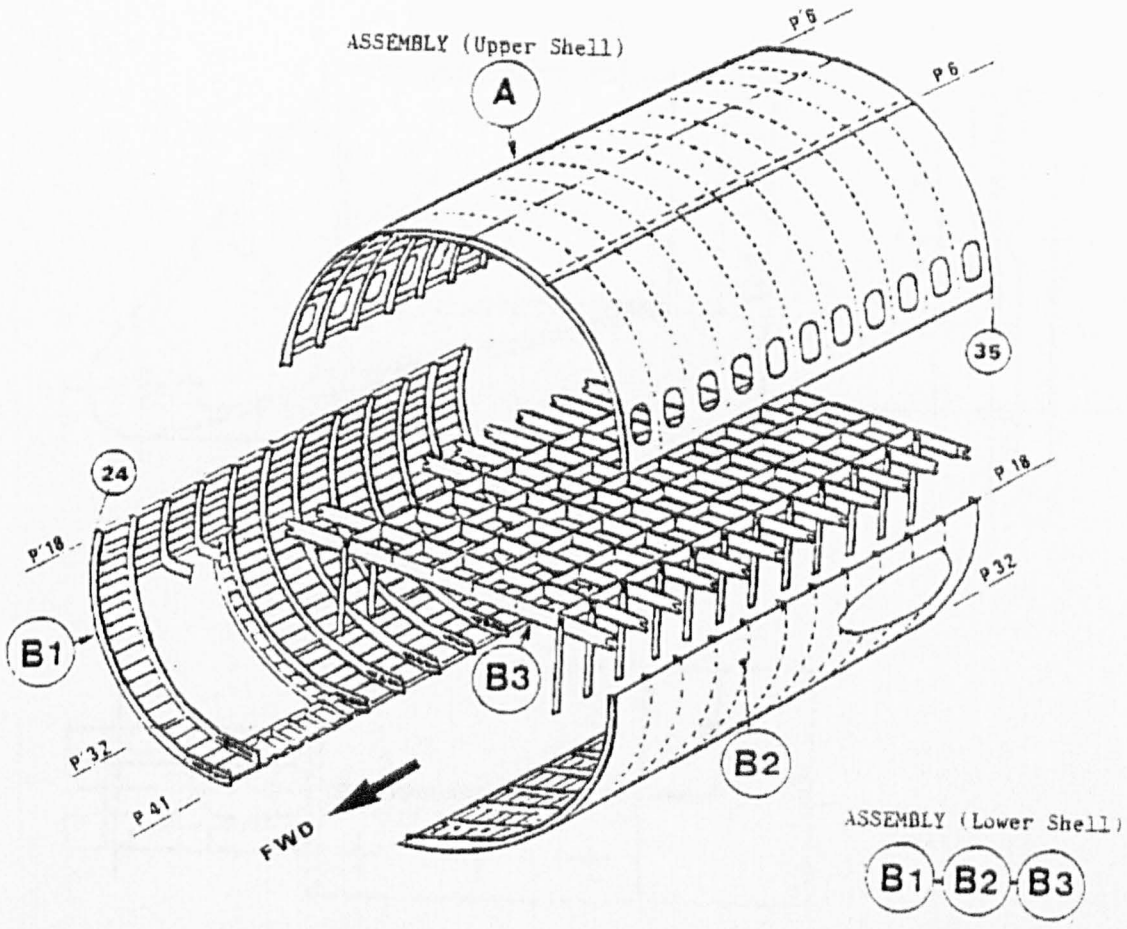


Figure 49: work package for forward fuselage

Because this cost estimating procedure uses elements of **Detailed Cost Estimates Based On Workpackages** (see chapter 3.2.9) (WBS, technical data, cost database) and elements of **Parametric Cost Estimates** (see chapter 3.2.11) (MCPLXS as CER's), this method can not be categorised as any particular cost estimating method. Rather this method can be positioned between these techniques and should be implemented between the phases B and C of Figure 13 on page 57. This method of cost estimation has advantages for DASA only because all accessible cost information can be used while an estimate is only applied when necessary.

7.3 Hogout Problem

Some components in the aeroplane industry are subject to extremely high performance requirements (e.g. metal fittings, brackets, trusses). For weight reasons many of these parts are made of aluminium. Though aluminium is very

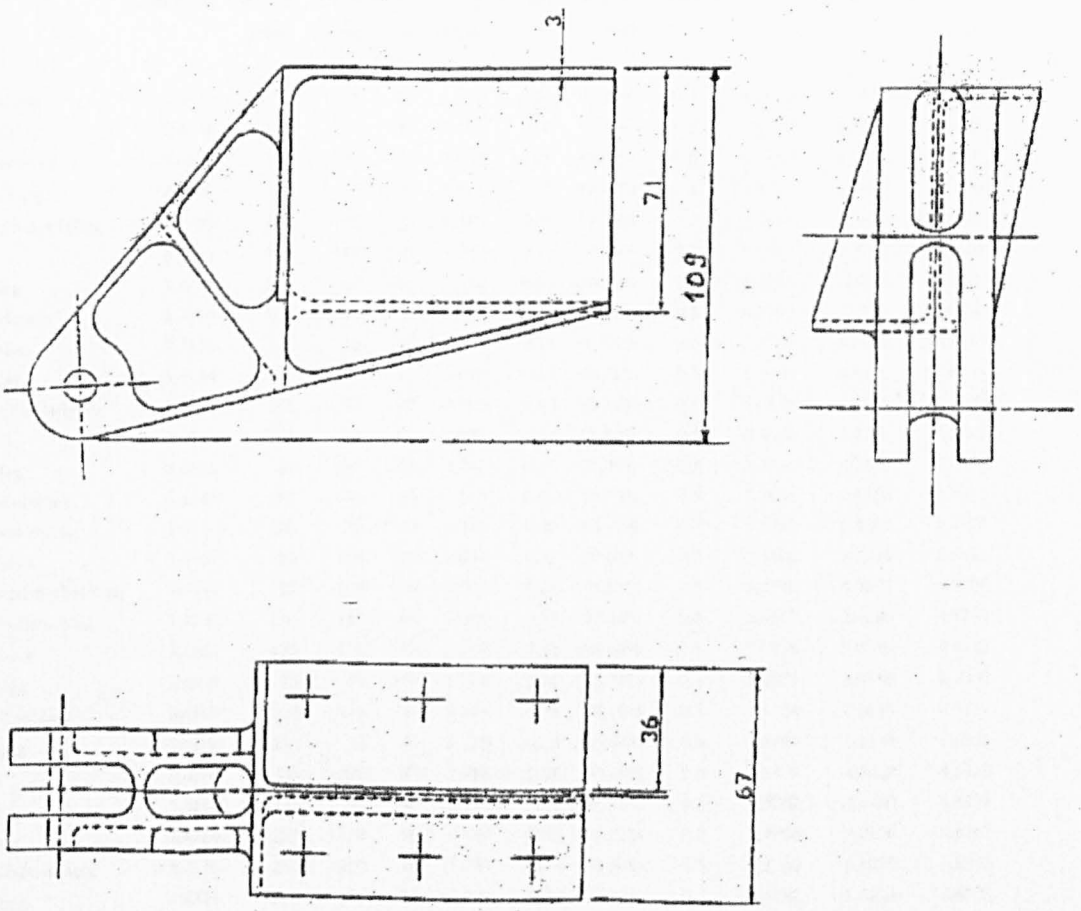


Figure 50: example for hogout component

light, it is also a very brittle metal and the bending or welding of aluminium leads to tensions and weakness inside the material and then the component could not fulfil its performance requirements anymore. To fulfil these requirements the components are milled out of aluminium blocks.

Depending on the component's geometry up to 98% of the aluminium block are milled off (see Figure 50) to produce the required shape. Because DASA calls this extensive milling process "Hogout" this expression will be used in the following chapters. The costs of these hogout components are determined by

- the material cost (of the aluminium block).
- the milling costs (the more material is removed the more expensive the component becomes).
- the geometry of the component (the more complex the geometry the more often the tools of the CNC milling-machine need to be changed or the more often the

no.	name	Material	length [mm]	width [mm]	dept h [mm]	raw weight	weight	Hogout [%]	Prec.	MCPLXS + H.	MCPLXS cal	MCPLXS - H.
1	Y-Beschlag	3.4364	75	60	80	1.008	0.05	95.04%	0.3	5.5692	4.3840	4.8068
3	X-Beschlag	3.4364	120	110	90	3.326	0.06	98.20%	0.5	5.6112	6.3800	4.6190
5	XY-Beschlag	3.4144	150	110	90	4.158	0.08	98.08%	0.5	5.6156	6.3390	4.6226
7	X-Beschlag	3.4364	80	80	90	1.613	0.09	94.42%	0.3	5.6010	5.9790	4.8076
10	Fairing Befestigung	3.4364	140	53	50	1.039	0.09	91.34%	0.5	5.1882	4.3640	4.6215
11	Halter	3.4364	100	160	100	4.480	0.11	97.54%	0.3	5.7493	5.7740	4.8104
12	Beschlag	3.4364	140	80	100	3.136	0.12	96.17%	0.5	5.4174	4.7010	4.6215
13	Tab Führung	3.4364	130	30	32	0.347	0.13	62.80%	0.5	4.7539	5.2780	4.6202
15	Kupplung	3.4364	125	70	80	1.960	0.13	93.37%	0.5	5.2784	4.5750	4.6196
16	Beschlag	3.4364	105	105	80	2.470	0.14	94.33%	0.3	5.4754	4.6190	4.8111
17	Fairing Befestigung	3.4364	140	75	32	0.935	0.14	85.12%	0.5	5.0606	4.6750	4.6215
18	Beschlag	3.4364	340	66	80	5.027	0.17	96.62%	0.5	5.4864	6.4770	4.6410
20	Beschlag	3.4364	80	115	80	2.061	0.17	91.75%	0.3	5.4141	4.7280	4.8076
21	Galleybeschlag	3.4144	130	95	90	3.112	0.19	93.89%	0.5	5.2864	4.6250	4.6202
23	Endrippe Außen	3.4364	520	65	40	3.786	0.20	94.72%	0.8	5.1943	5.4500	4.4832
24	Lagerbock	3.4364	230	110	100	7.084	0.21	97.04%	0.5	5.5102	5.4210	4.6311
27	Verriegelungsbeschlag	3.4144	100	100	90	2.520	0.21	91.67%	0.3	5.4746	4.9870	4.8104
28	Anschlußbeschlag	3.4144	130	130	90	4.259	0.24	94.36%	0.5	5.3017	5.4280	4.6202
29	Lagerbock	3.4364	155	120	80	4.166	0.25	94.00%	0.5	5.2936	5.0750	4.6232
30	Beschlag	3.4364	70	90	100	1.764	0.26	85.26%	0.3	5.2853	3.9710	4.8060
32	Hakenbeschlag	3.4364	105	134	90	3.546	0.28	92.10%	0.3	5.4754	4.9690	4.8111
33	Beschlag	3.4144	272	90	90	6.169	0.28	95.46%	0.5	5.3898	5.1180	4.6351
34	Hebel	3.4364	245	100	80	5.488	0.28	94.90%	0.5	5.3353	4.5420	4.6326
35	Stütze	3.4144	325	90	110	9.009	0.29	96.78%	0.5	5.5002	4.9440	4.6398
36	Stütze	3.4364	290	150	80	9.744	0.32	96.72%	0.5	5.4985	5.2120	4.6367
37	Anschlußbeschlag	3.4364	280	220	80	13.798	0.34	97.54%	0.5	5.5760	6.1270	4.6358
38	Beschlag	3.4364	300	165	100	13.860	0.34	97.55%	0.5	5.5782	5.4620	4.6376
39	Beschlag	3.4364	300	165	100	13.860	0.34	97.55%	0.5	5.5782	5.6990	4.6376
40	Beschlag	3.4144	205	170	120	11.710	0.35	97.01%	0.5	5.5152	5.9030	4.6286
42	Drift-Pin-Beschlag	3.4144	131	101	90	3.334	0.37	88.90%	0.5	5.0964	4.9580	4.6204
43	Endrippe Innen	3.4364	100	52	40	0.582	0.37	36.47%	0.3	4.9402	4.9450	4.8104
45	Beschlag	3.4144	131	101	90	3.334	0.37	88.90%	0.5	5.0964	4.9910	4.6204
46	Stützbeschlag	3.4144	160	80	80	2.867	0.37	87.10%	0.5	5.1002	5.1410	4.6238
47	Beschlag	3.4144	131	101	90	3.334	0.38	88.60%	0.5	5.0887	4.1790	4.6204
48	Hebel	3.4364	120	100	90	3.024	0.38	87.43%	0.3	5.3009	4.5450	4.8131
49	Anschlußbeschlag	3.4364	300	220	80	14.784	0.39	97.36%	0.5	5.5587	6.1670	4.6376
50	Stütze	3.4364	350	170	80	13.328	0.41	96.92%	0.5	5.5088	5.3640	4.6419
51	Stütze	3.4364	300	180	80	12.096	0.41	96.61%	0.5	5.4810	5.0460	4.6376
53	Hebellager hinten	3.4144	160	150	130	8.736	0.42	95.19%	0.5	5.3766	5.0180	4.6238
54	Hebel	3.4364	190	120	80	5.107	0.43	91.58%	0.5	5.2075	4.6840	4.6271
55	Stützbeschlag	3.4144	140	116	120	5.457	0.43	92.12%	0.5	5.2012	5.4130	4.6215
56	Stützbeschlag	3.4144	140	116	120	5.457	0.43	92.12%	0.5	5.2012	5.4130	4.6215
58	Stützbeschlag	3.4144	140	116	120	5.457	0.43	92.12%	0.5	5.2012	5.4170	4.6215
59	Beschlag	3.4144	250	200	130	18.200	0.47	97.42%	0.5	5.5520	5.8010	4.6330

Table 25: MCPLXS of Hogout components (part 1)

component has to be clamped in different positions and the longer the production process lasts).

Generally it can be said that the longer the milling process lasts the more expensive the component will be. Or: **The lighter the component the more expensive it is!**

If the costs of hogout elements are estimated with the PRICE model, weight and complexity are the main influencing factors. According to the formula in chapter 5.2 the MCPLXS is determined by Machinability, Maturity, Platform, Precision, Length and Number of Parts. Since all components are made of Aluminium (two very similar materials are used) and because their assembly difficulties are almost equal and the number of fabricated parts is one, their Maturity (3), Machinability (160), Platform (1.7) and number of parts (1) for all components remain the same. Merely the Material has two different but still very similar values (3.4144 and 3.4364) which have no effect on the Machinability and therefore this value can be neglected. MCPLXS varies due to a different Precision (between 0.3 and 2) or Length (longest side of length, depth or width). Because Length and Precision play only an inferior role in the determination of MCPLXS the main contribution to calculate the costs is given by the weight.

Now the PRICE model is set up in such a manner that an increasing weight (or volume) of the component leads to higher production costs. For hogout components this connection is exactly the other way around! The higher the hogout the lighter the component becomes and the more expensive the production will be. The PRICE model fails because it underestimates the production costs for parts which have a relatively high hogout.

To solve this problem, PRICE introduced a correction value named slug factor which directly influences the complexity. The hogout for the determination of the complexity has the following influence:[87]

$$\text{Slug Factor} = \left(\frac{\text{Weight of Slug}}{\text{Weight of Finished Item}} \right)^{.05}$$

If more than 10% of the slug weight is machined away this factor is multiplied with the original complexity. Because this value was not sufficient to handle the hogout problem for DASA, PRICE introduced another correction factor, which directly influences the costs according to the following formula:

[87] PRICE H Reference Manual, PRICE Systems, Moorestown, New Jersey, 1988, 20-13

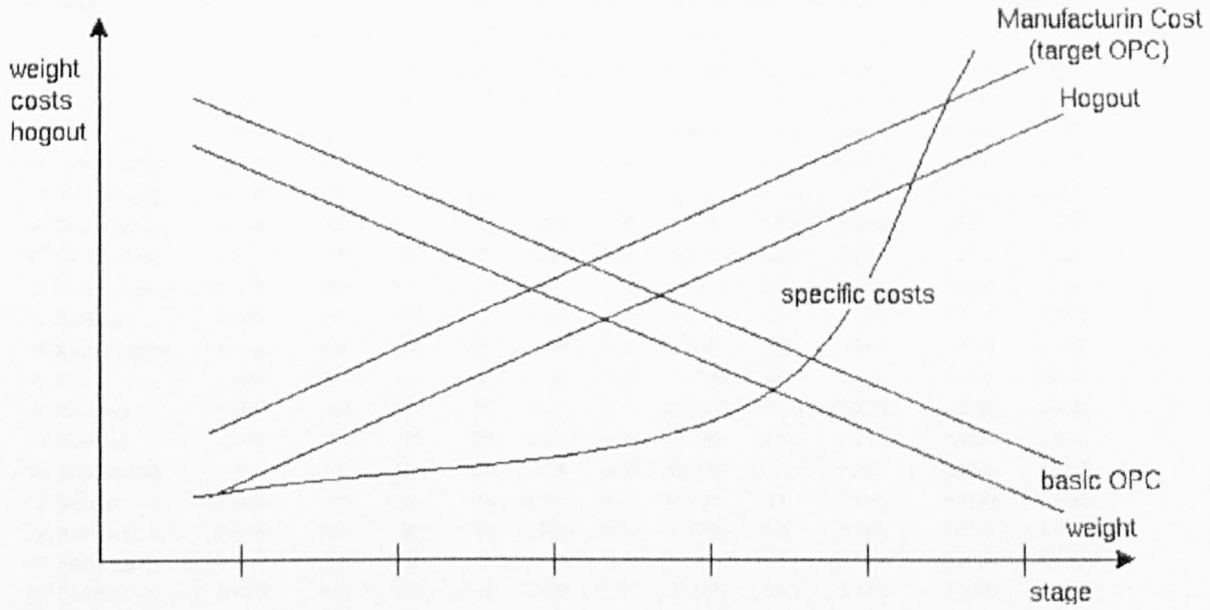


Figure 51: Influence of Hogout on basic OPC, target OPC and specific cost

target OPC = basic OPC + (basic OPC * hogout * K)^{KEXP}

OPC (only piece cost) describes the costs for the production of a single component. Economies of scale and learning curves are not considered. To calculate the target OPC the basic OPC are corrected with the hogout factor (the higher the hogout the more costs are added). K and KEXP can be chosen and adjusted by the user until the target costs have been reached. To get a number of target OPC values DASA milled an aluminium block in small steps down to the required component. For each of these steps the hogout and the weight of the component and the manufacturing time were determined and multiplied with the relevant production costs.

If these values are entered in a co-ordinate grid the result might look similar to Figure 51. According to the original PRICE model the decreasing weight of a component leads to decreasing basic OPC (basic OPC line) although in reality the actual costs increase (target OPC line). In fact the specific costs of a component (Deutsche Mark per Kilogramm) increase exponentially. Based on the curves of basic OPC and hogout K and KEXP are adjusted in such a manner until the equation represents the target OPC curve.

no.	Name	Material	length [mm]	width [mm]	depth [mm]	raw weight	weight	Hogout [%]	Prec.	MCPLXS + H.	MCPLXS cal	MCPLXS - H.
60	Beschlag	3.4144	250	200	130	18.200	0.47	97.42%	0.5	5.5520	6.1660	4.6330
62	Beschlag	3.4144	250	200	130	18.200	0.47	97.42%	0.5	5.5520	5.8010	4.6330
63	YZ-Riegel	3.4364	153	102	100	4.370	0.48	89.02%	0.5	5.1073	4.6110	4.6230
64	Tab Anschluss	3.4364	245	54	60	2.223	0.49	77.95%	0.5	4.9007	5.2750	4.6326
65	Stützbeschlag	3.4144	147	116	120	5.729	0.51	91.10%	0.5	5.2054	4.7250	4.6223
66	TAB-Beschlag	3.4364	290	160	80	10.394	0.55	94.71%	0.5	5.3452	4.9480	4.6367
67	Tab Beschlag	3.4364	290	160	80	10.394	0.55	94.71%	0.5	5.3452	5.2380	4.6367
69	Gabelhalterung	3.4364	200	130	80	5.824	0.61	89.53%	0.5	5.1608	4.3450	4.6281
70	Betätiger	3.4364	284	92	90	6.584	0.65	90.13%	0.5	5.1958	5.2030	4.6362
71	Klappenbeschla	3.4364	490	190	100	26.068	0.65	97.51%	0.8	5.3819	5.8250	4.4811
73	TLU-	3.4364	405	170	80	15.422	0.66	95.72%	0.8	5.2193	5.3590	4.4747
74	Beschlag	3.4144	360	90	90	8.165	0.66	91.92%	0.5	5.2369	4.6150	4.6427
75	Beschlag	3.4144	185	130	120	8.081	0.68	91.58%	0.5	5.2102	5.4330	4.6265
76	Stützbeschlag	3.4144	140	160	120	7.526	0.70	90.70%	0.5	5.1963	5.6500	4.6215
77	Segment	3.4144	310	300	110	28.644	0.71	97.52%	0.5	5.5769	6.7120	4.6385
78	Beschlag	3.4144	410	100	90	10.332	0.72	93.03%	0.8	5.0852	4.9750	4.4751
79	FAN-Lager	3.4364	350	400	80	31.360	0.75	97.61%	0.5	5.5844	5.8180	4.6419
80	Beschlag	3.4364	490	200	100	27.440	0.77	97.19%	0.8	5.3456	5.3790	4.4811
81	Beschlag	3.4144	325	100	90	8.190	0.78	90.48%	0.5	5.1858	4.3660	4.6398
82	Nasenbeschlag	3.4364	200	100	100	5.600	0.80	85.71%	0.5	5.0822	4.9390	4.6281
84	Schloßbeschlag	3.4364	300	250	100	21.000	0.84	96.00%	0.5	5.4363	5.3960	4.6376
85	Schloßbeschlag	3.4144	300	250	110	23.100	0.85	96.32%	0.5	5.4587	5.4280	4.6376
86	Schloßbeschlag	3.4364	320	240	100	21.504	0.86	96.00%	0.5	5.4449	5.2650	4.6393
87	Schloßbeschlag	3.4144	300	220	110	20.328	0.87	95.72%	0.5	5.4126	5.2750	4.6376
88	Antriebsbeschla	3.4364	150	100	90	3.780	0.88	76.72%	0.5	4.9246	4.0980	4.6226
90	Oberer Beschlag	3.4144	210	130	140	10.702	0.90	91.59%	0.5	5.2240	5.5300	4.6291
92	Schloßbeschlag	3.4144	290	250	110	22.330	0.90	95.97%	0.5	5.4289	5.2980	4.6367
93	Hebelwelle	3.4364	380	250	90	23.940	0.91	96.20%	0.5	5.4593	5.2590	4.6443
94	Spantkupplung	3.4364	250	190	90	11.970	1.05	91.23%	0.5	5.2093	4.9730	4.6330
96	Traverse	3.4364	375	110	80	9.240	1.11	87.99%	0.5	5.1224	4.6870	4.6439
97	Traverse	3.4364	430	110	80	10.595	1.19	88.77%	0.8	4.9746	4.7710	4.4766
98	Beschlag	3.4144	280	170	100	13.328	1.21	90.92%	0.5	5.1947	5.6520	4.6358
100	Traverse	3.4364	450	110	80	11.088	1.23	88.91%	0.8	4.9670	4.7960	4.4781
102	Beschlag	3.4144	290	130	130	13.723	1.25	90.89%	0.5	5.2076	4.5700	4.6367
103	Halterung	3.4144	290	140	130	14.778	1.70	88.50%	0.5	5.1390	4.5080	4.6367
106	Beschlag	3.4144	300	150	130	16.380	1.73	89.44%	0.5	5.1536	5.0720	4.6376
107	Führungsschien	3.4144	580	190	140	43.198	1.95	95.49%	0.8	5.2258	5.5570	4.4874
109	Rippe 13	3.4364	750	200	80	33.600	2.00	94.05%	0.8	5.1671	5.3710	4.4983
110	Beschlag	3.4364	515	74	80	8.537	2.17	74.58%	0.8	4.7474	4.6830	4.4829
112	Antriebshebel	3.4364	610	175	90	26.901	2.19	91.86%	0.8	5.0687	5.0050	4.4894
113	Traverse	3.4364	780	175	90	34.398	2.39	93.05%	0.8	5.1202	5.3370	4.5001
114	Holm	3.4364	3050	85	55	39.925	2.76	93.09%	2	4.8777	5.0380	4.2826
115	Traverse	3.4144	700	130	130	33.124	4.80	85.51%	0.8	4.9113	4.3740	4.4952
116	Mittelträger	3.4144	1110	240	130	96.970	6.44	93.36%	1.2	4.9939	5.7810	4.3756

Table 26: MCPLXS of Hogout components (part 2)

The correction values K and KEXP become invalid if

- a different material is used which will change the production time and production costs
- the component is produced on a different grinder or milling-machine
- the geometry of the component is more or less complex; its production can be easy (cheap) or difficult (expensive).

Strictly speaking K and KEXP are only valid for a certain component that is produced on a certain milling-machine with the same grinders. As soon as one of the variables changes (and at DASA this is often the case) new values need to be determined for K and KEXP which is impractical.

Table 25 and Table 26 show some samples of hogout components that are made of aluminium blocks. The complexity in the right column MCPLXS - H (- H. stands for "no consideration of Hogout") is the result of the classical PRICE model. MCPLXS + H considers the hogout. In combination with the weight both types of complexities lead to inaccurate estimates.

In effect this means that to calculate costs for articles with hogout, PRICE applies a different complexity parameter, which accounts for the hogout, and then modifies the cost estimates produced by another factor which also includes an element to take account of the hogout.

To calculate the correct complexity for each component in Table 25 and Table 26 the production costs were determined and then traced backwards to derive a calibrated complexity (MCPLXS - cal). The PRICE software program offers a specific tool for the trace called ECIRP (PRICE backwards).

In the following chapters it will be examined to which extent a neural network is capable of determining MCPLXC - cal by using inputs like Material, Length, Width, Depth, Raw Weight, Weight, Hogout and Precision. Is a neural network capable to represent a functional relationship that has a higher quality than the values determined by the PRICE model?

To exclude parameters that might influence MCPLXS - cal without being listed as input values the following has been considered for the selection of components:

- All components have been manufactured on similar milling-machines (single-spindle-milling-machine) in order to be based on similar production costs.

- All components are made of the same aluminium. Therefore it can be assumed that for a similar geometry the same tools (rotary grinders) and rotating and progressing speeds will be used.
- All components have been manufactured during the same period of time in order to have similar economic conditions.
- For all components identical lot sizes have been produced in order to keep the costs for tool changes constant.

Although the components were comparable there can still be some scattering in the data: One reason could be that during the manufacturing process the milling-machine comes to an unexpected stop, for example, due to a broken rotary grinder. This delay increases the production time and the production costs. DASA's experience shows that these additional expenses do not exceed 8%.[88] Considering that the relationship between costs and MCPLXS is not necessarily linear a data analysis shows that this scattering never exceeds 6% for the same components.

In the following sections the data will be normalised in different ways and the configuration of the networks will vary. Both software programs Qnet and Nnmodel will be applied. The analysis procedure is analogous to chapter 5.2 .If any set up should differ from the corresponding tests in chapter 6 then this will be explained further.

7.4 NNModel for the Hogout problem

7.4.1 Preparation

In this test NNModel will be used to determine three types of complexities. The program is set-up according to chapter 5.1 (The software NNMODEL). Again the results are five neural models and five test matrices that have been tested with their corresponding model. The only difference between chapter 5 (Multi-Layer

[88] Telephone call between Mr. Hübner and author on November, 12th 1997 11.30

MCPLXS – H. measured	P MCPLXS - H predicted	R MCPLXS - H residual
4.474691	4.473596	0.001095
4.48107	**4.473536**	**0.007533**
4.48107	**4.473701**	**0.007369**
4.48107	**4.479446**	**0.001624**
4.489364	**4.494316**	**-0.004952**
4.489364	**4.497313**	**-0.007949**
4.619629	4.62234	-0.00271
4.620249	**4.628334**	**-0.008085**
4.620372	**4.626222**	**-0.00585**
4.62146	**4.628211**	**-0.006751**
4.62146	**4.629506**	**-0.008046**
4.62146	**4.631815**	**-0.010355**
4.62146	**4.631815**	**-0.010355**
4.62146	**4.634426**	**-0.012966**
4.62146	**4.645766**	**-0.024306**
4.62146	**4.64635**	**-0.02489**
4.622287	4.629014	-0.006726
4.623783	**4.630459**	**-0.006676**
4.623783	**4.631138**	**-0.007355**
4.623783	**4.635849**	**-0.012066**
4.628105	4.642135	-0.01403
4.62862	4.636813	-0.008193
4.62913	4.642021	-0.012891
4.631122	**4.634059**	**-0.002937**
4.631122	**4.637192**	**-0.00607**
4.633047	**4.637224**	**-0.004177**
4.633047	**4.639218**	**-0.006171**
4.633047	**4.642731**	**-0.009684**
4.633047	**4.642962**	**-0.009915**
4.635093	4.642504	-0.007411
4.636182	4.642176	-0.005994
4.636719	**4.639873**	**-0.003153**
4.636719	**4.643184**	**-0.006465**
4.637605	**4.644464**	**-0.006859**
4.637605	**4.645932**	**-0.008327**
4.637605	**4.648191**	**-0.010586**
4.63977	4.645415	-0.005646
4.806042	4.801698	0.004344
4.806818	**4.795312**	**0.011506**
4.806818	**4.801388**	**0.00543**
4.807574	4.793025	0.014549
4.810426	**4.792469**	**0.017957**
4.810426	**4.807561**	**0.002865**

	absolute	relative
mean Error of R-MCPLXS – H.	0.008	0.182%
relative max. Error of R-MCPLXS – H.	0.025	0.539%
absolute max. Error R-MCPLXS – H.	0.025	0.539%

Table 27: Results of MCPLXS - H with NNMODEL

Perceptron to determine values of the complexity of the PRICE-Model) and this test is that other input parameters are used.

As well as Precision other variables such as Length, Width, Depth and Weight are included. Platform, Maturity and Machinability are not applied as they remain the constant at 1.7, 3 and 160 respectively. And so they cannot reveal any useful information to the network.

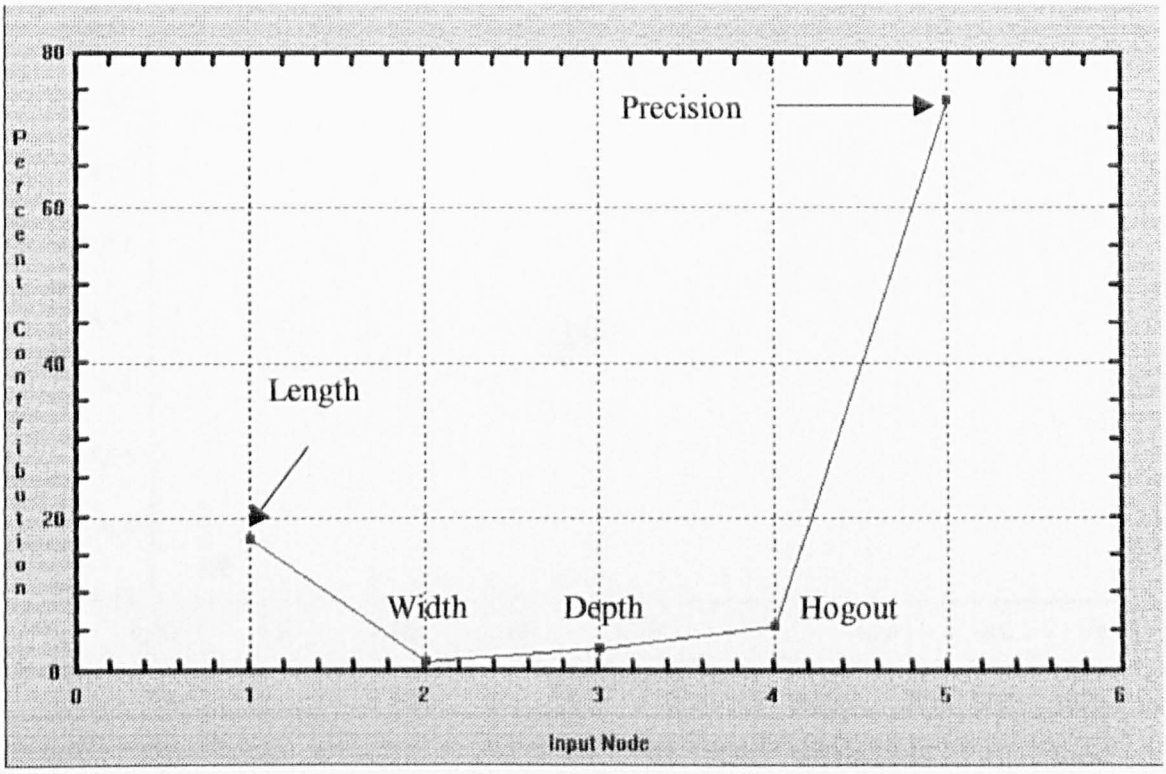


Figure 52: Percent Contribution for MCPLXS - H of inputs

7.4.2 Results

For MCPLXS - H the network performs extremely well. Table 27 shows that the relative mean error is only 0.182% (absolute R_MCPLXS - H is 0.008). The maximum error is 0.539% (R_MCPLXS - H is 0.025). These results are better than those of chapter 5.2. This is not really surprising because according to chapter 5.2 the Platform, Precision, Number of parts, Machinability, Maturity and Length determine MCPLXS.

In this test most values are constant and only Length and Precision influence the complexity because the other input parameters (Width, Depth, Weight) are not considered in the mathematical calculation of MCPLXS - H. As it has already been mentioned in chapter 7.4.1 the material is not considered as an input parameter since it is already built into the Machinability factor.

Figure 52 shows how much each input contributes to the output. The fifth input (Precision) and the first input (Length) determine the output (MCPLXS – H) by

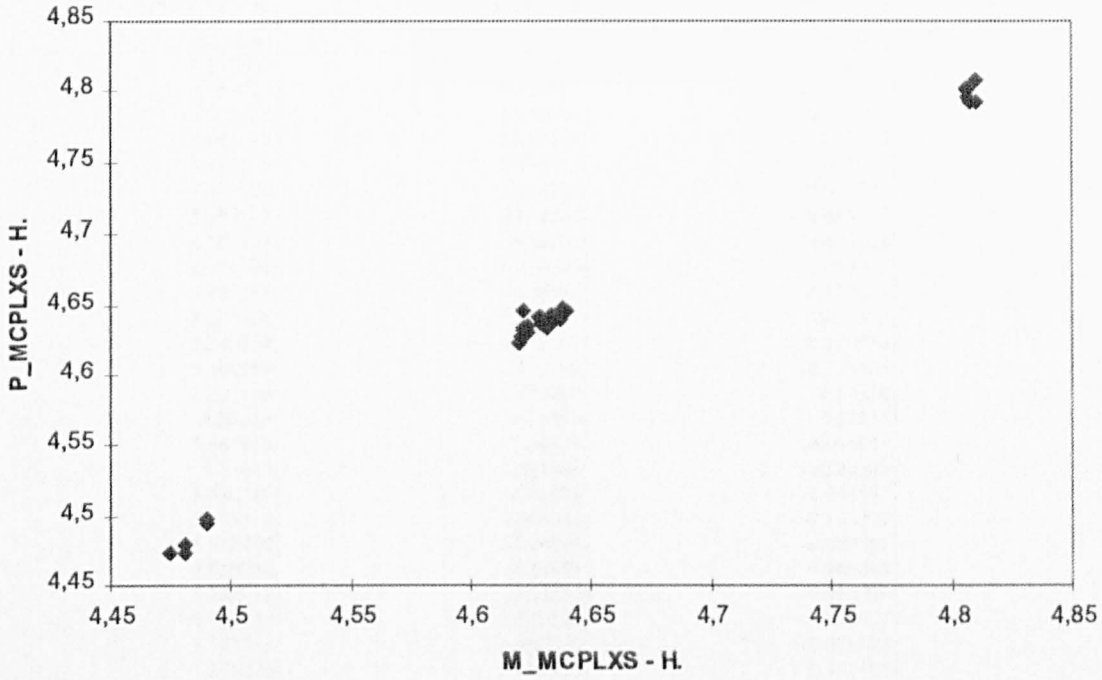


Figure 53: measured versus predicted MCPLXS - H

almost 80% and 20% respectively. The other variables have hardly any influence on the output at all. With only two main variables in the formula the emulated equation becomes much simpler than the one mentioned in chapter 5.2. Therefore the error of the neural net becomes smaller.

Another reason for having a better performance than in chapter 5.2 is that the determined complexities are very similar. Figure 53 shows that there are three clusters of the complexities which are used for the training. Since the values for MCPLXS - H can only be located within one of these clusters and because the neural net can concentrate on these very limited areas the performance within these ranges will be particularly good. Therefore it is very likely that the error within these clusters is lower than the error of a neural network where the data is evenly spread as it is in chapter 5.2.

The performance of the second neural net where the manufacturing complexity with hogout (MCPLXS + H) is determined is not as good as the one for MCPLXS - H. Table 28 shows that the relative mean error is 2.134% (absolute R_MCPLXS - H is 0.111). The maximum error is 14.106% (R_MCPLXS - H is 0.688).

MCPLXS – H. measured	P MCPLXS - H predicted	R MCPLXS - H residual
4.877653	5.565707	-0.688055
4.967021	5.027063	-0.060042
5.096413	**5.253608**	**-0.157195**
5.096413	**5.275098**	**-0.178685**
5.122374	5.136372	-0.013998
5.139028	5.118254	0.020774
5.153623	**5.134869**	**0.018754**
5.153623	**5.14037**	**0.013253**
5.194731	5.216723	-0.021992
5.196255	5.320769	-0.124514
5.225821	5.321611	-0.09579
5.293569	5.413007	-0.119438
5.300866	5.454292	-0.153426
5.301689	5.443969	-0.14228
5.33533	5.46706	-0.13173
5.345608	5.44223	-0.096622
5.37662	5.505005	-0.128385
5.459341	5.500938	-0.041597
5.475374	5.608595	-0.133221
5.510218	**5.570004**	**-0.059786**
5.510218	**5.57091**	**-0.060692**
5.552018	5.614224	-0.062206
5.576919	5.616629	-0.03971
5.584393	5.654266	-0.069873
5.611226	5.469125	0.142101

	absolute	relative
Mean Error of MCPLXS + H	0.111	2.134%
relative max. Error of MCPLXS + H	0.688	14.106%
absolute max. Error of MCPLXS + H	0.688	14.106%

Table 28: Results of MCPLXS + H with NNMODEL

Compared with the results of chapter 5.2 this error is still acceptable. Possible reasons for this slightly lower performance are:

1. The emulated formula is more complex because another influencing parameter has to be considered (Hogout). The formula in chapter 7.3 shows the influence of the hogout and Figure 54 indicates that the hogout influences MCPLXS + H by almost 30%. Actually the absolute or relative hogout is not directly implemented as input parameter. The weight of the finished item has been used instead because by calculating the raw weight with the help of the dimensions (length, width and depth) and subtracting the weight of the finished item the hogout can be calculated. Since weight and hogout are extremely dependent on each other the neural net can go without one of these values. As for the determination of MCPLXS - H Length and Precision are the other main inputs.

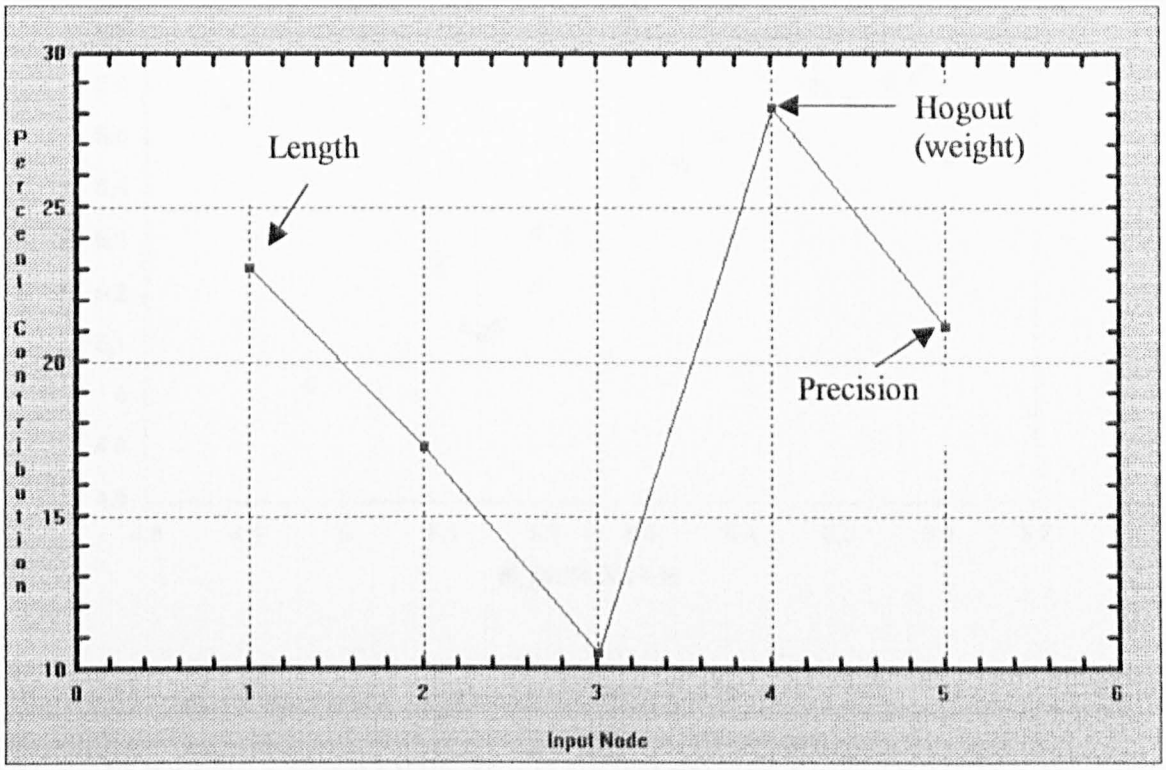


Figure 54: Percent Contribution for MCPLXS + H of inputs

2. Compared to MCPLXS - H the present complexities are more evenly spread within their data range (see Figure 55). This makes it more difficult for the neural net to train well because it can not concentrate on particular clusters.

Figure 55 represents graphically the predicted and measured data for MCPLXS + H in Table 28 and it can be seen that the first pair is an outlier.

The NNModel greatly overestimates the complexity for this data-set because the corresponding Length is set to 3050 which is very long compared to the other values which all lie between 70 and 1110. Since the neural net is quite sensitive to Length an "overreaction" is quite understandable. If the outlier is neglected the relative mean error drops to 1.57% and the maximum error is only 3.506%. This result is very comparable to chapter 5.2.

In the training data some data-sets can occur several times (see Table 25 and Table 26). Therefore they have the same calculated complexity (MCPLXS + H and MCPLXS - H). This redundant information hardly affects the training or performance of the neural net because these data-sets are simply trained more often than those which occur only once. Data-set no. 16 and no. 32 (see Table 25)

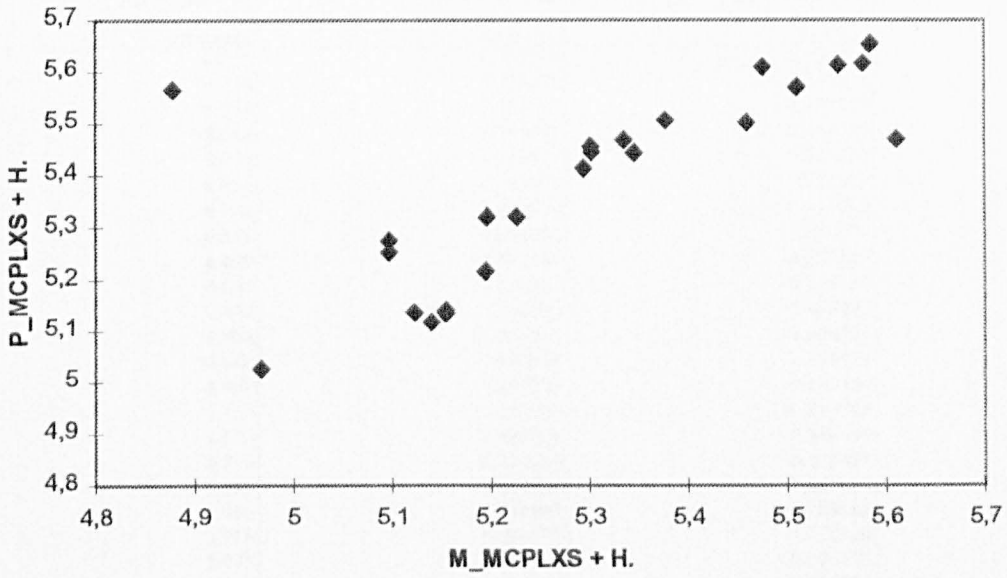


Figure 55: measured versus predicted MCPLXS + H

have identical MCPLXS + H and MCPLXS - H but different input parameters. They are treated as different data-sets and do not disturb the training.

According to Table 29 for MCPLXS - cal the relative mean error is 7.972% (absolute R_MCPLXS - H is 0.406). The maximum error is 21.518%. Figure 56 displays the contribution of each input variable for the output. It is surprising that the hogout is not the main variable for the determination of the MCPLXS – cal. The Length has a slightly higher influence on the network. Also the contributions for MCPLXS – cal are very similar to the contributions for MCPLXS + H. This can be interpreted as a first hint that the equation applied in PRICE represents the influence of each CER reasonably well.

Considering that these values do not originate from a mathematical function the networks performance is not too bad. Now the question arises, if this network has a better performance than MCPLXS + H and whether it should be used instead of the PRICE model. To answer this question all MCPLXS + H are compared with their corresponding MCPLXS - cal The results can be seen in Table 30. A relative mean error of only 7.591% (absolute R_MCPLXS - H is 0.402) and a maximum error of 24.867% shows that the performance of the neural network and the PRICE-model is almost identical. The next section will give a closer insight into the error of both types of models.

MCPLXS - cal measured	MCPLXS - cal predicted	MCPLXS – cal residual
4.098	4.787564	-0.689564
4.374	5.315212	-0.941212
4.542	**5.319787**	**-0.777787**
4.542	**5.383671**	**-0.841671**
4.619	5.152577	-0.533577
4.683	4.563971	0.11903
4.796	5.209992	-0.413992
4.939	**4.888626**	**0.050374**
4.939	**5.012149**	**-0.073149**
4.944	5.682167	-0.738167
4.948	5.414205	-0.466205
4.958	**5.056367**	**-0.098367**
4.958	**5.202079**	**-0.244079**
4.969	5.164799	-0.195799
4.973	5.187041	-0.214041
4.975	**5.331419**	**-0.356419**
4.975	**5.404494**	**-0.429494**
5.005	**5.268828**	**-0.263828**
5.005	**5.290088**	**-0.285088**
5.018	6.096589	-1.078588
5.075	5.272663	-0.197663
5.141	**4.934148**	**0.206851**
5.141	**5.046308**	**0.094692**
5.212	5.581754	-0.369754
5.238	5.423175	-0.185175
5.275	**4.895454**	**0.379546**
5.275	**5.918465**	**-0.643465**
5.278	5.236977	0.041023
5.298	6.079223	-0.781223
5.337	5.296154	0.040846
5.371	**5.199326**	**0.171674**
5.371	**5.391578**	**-0.020578**
5.417	5.370216	0.046783
5.428	6.137373	-0.709373
5.65	5.53475	0.11525
5.699	5.965871	-0.266871
5.801	**6.260792**	**-0.459792**
5.801	**6.261063**	**-0.460063**
5.818	6.110426	-0.292426
5.903	6.075809	-0.17281
5.979	4.831769	1.147231
6.167	5.701985	0.465014
6.339	5.483844	0.855156
6.477	5.263063	1.213937
6.712	**6.194928**	**0.517072**
6.712	**6.24968**	**0.46232**

	absolute	relative
Mean Error of MCPLXS – cal	0.406	7.927%
relative max. Error of MCPLXS – cal	0.941	21.518%
absolute max. Error of MCPLXS – cal	1.214	18.742%

Table 29: Results of MCPLXS - cal with NNMODEL

It will be the task of the following sections to develop a neural net that beats the results of Table 30. Therefore the influence of each input parameter on the output and other dependencies will be discussed later.

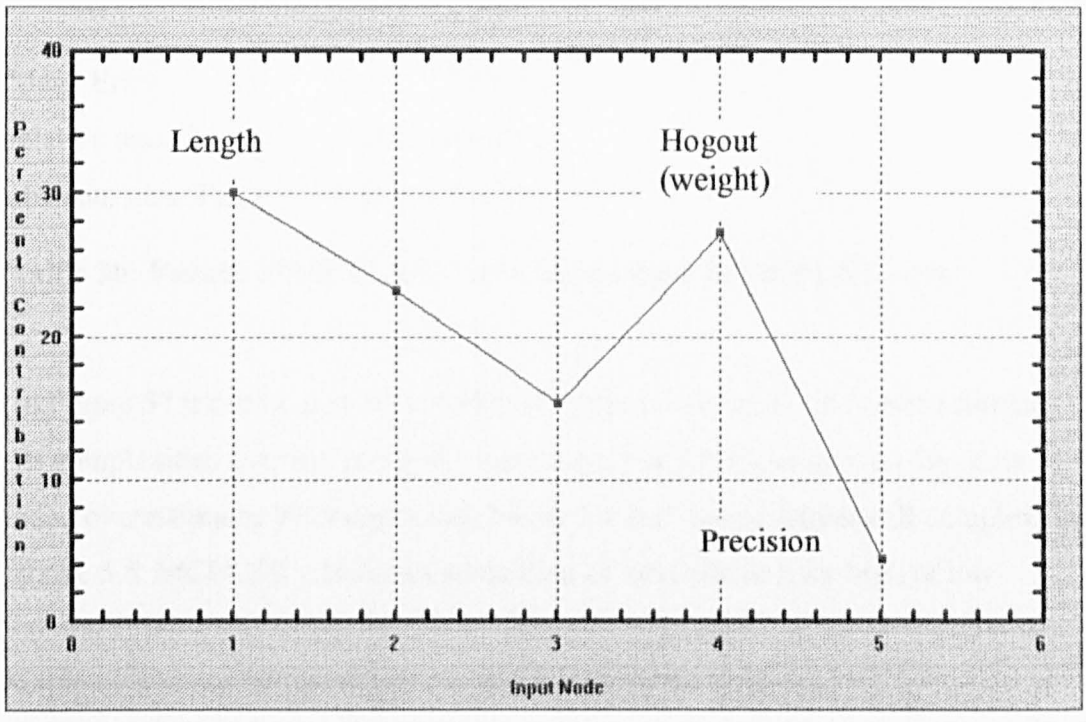


Figure 56: Percent Contribution for MCPLXS - cal of inputs

It should be mentioned that different inputs result in the same output of MCPLXS - cal, namely 5.275, 5.413, 5.428 and 5.801. But in this randomly chosen test only for M_MCPLXS - cal = 5.275 two different data-sets have been used (for any other identical M_MCPLXS the same data-set has been applied). Compared to the other complexities for this value the two P_MCPLXS are very different (0.379546 and -0.643465). Because the neural net has very different results for these two data-sets it can be assumed that two different inputs have been used which in fact is the case.

7.4.3 Errors

Although the errors for the complexity values of the PRICE-model (MCPLXS + H) and those of the neural network are almost identical there are still significant differences. Figure 57 shows how the error of both models is distributed. The x-axis displays the aimed values for both models (MCPLXS - cal) while the output of each corresponding MCPLXS - cal can be seen from the y-axis. If the output would be equal to the aimed value the dot would be located on the 45° line. Points above this line represent overestimates and points below this line represent underestimates by the models.

	relative	absolute
Mean Error	0.402	7.591%
relative max. Error	1.314	24.867%
absolute max. Error	1.314	24.867%

Table 30: Results of MCPLXS + H in comparison to MCPLXS – cal

In Figure 57 it can be seen that NNModel quite evenly over- and underestimates its complexities over the complete data range. The PRICE-model on the other hand overestimates all complexities below 5.1 and underestimates all complexities above 5.5. MCPLXS + H shows some kind of inert reaction for high or low complexities while NNModel reacts quite flexibly. Again the following chapters will give a closer insight about the distribution of the errors and the influence of each input parameter on the output.

7.4.4 Conclusion

Again in this chapter a neural network was able of representing a mathematical equation for MCPLXS. For the determination of MCPLXS - cal which is not based on an equation the results were not as good as for MCPLXS - H and MCPLXS + H but still the neural net has achieved a quality that is comparable to a mathematical model that is currently used by DASA.

Since the neural network of this test is almost equivalent to the network that has been used in chapter 5.1 (The software NNMODEL) the question occurred, whether a different set-up of the model could improve the results. Therefore the author experimented with some different set-ups in order to observe any changes affecting the performance of the network. But no significant improvement of the model could be reached.

By looking at the influence of the input parameters on the output and other dependencies within the data and by considering the distribution of the errors it will be the task of the following sections to develop a neural net that exceeds the quality of the PRICE-model.

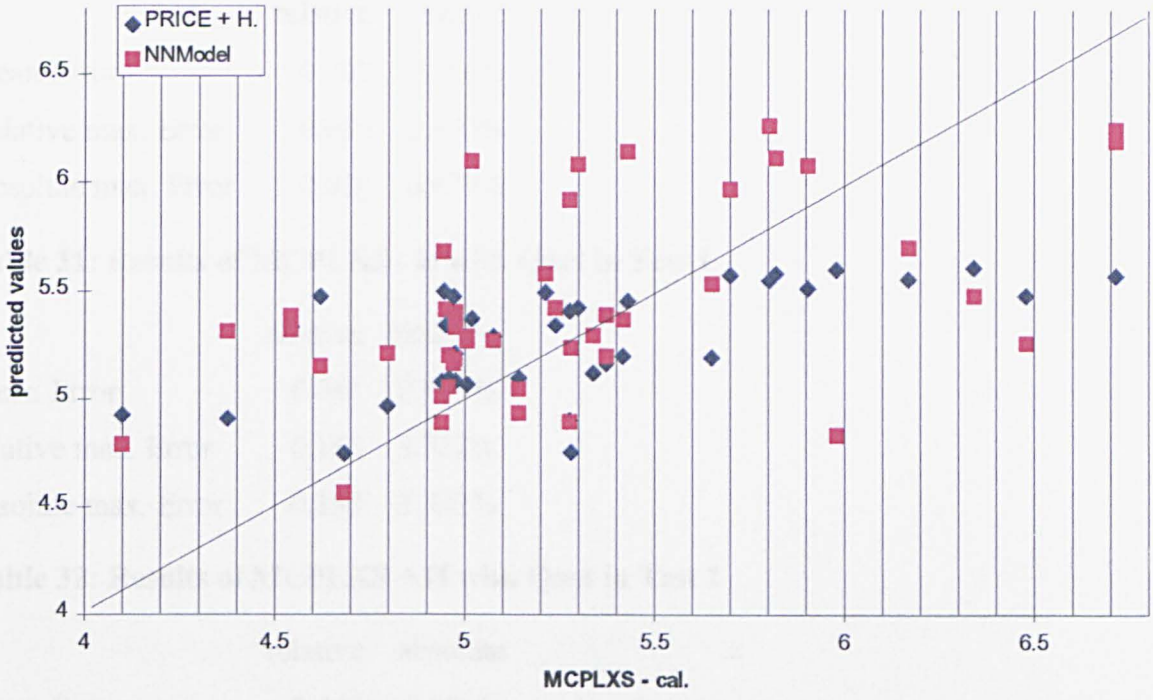


Figure 57: comparison of PRICE + H. and NNModel

7.5 Test 1 for the Hogout problem

7.5.1 Preparation

As in to Test 1 (chapter 6.2) a fully connected network with 5 neurons in the hidden layer was established. About 10% of the data was randomly excluded to test the performance of the neural models and again 5 networks were created to gather sufficient testing data.

7.5.2 Results

Table 31 shows the performance of the networks for the determination of MCPLXS - H. A relative mean error of 0.156% and a maximum error of 0.670% shows that Qnet performs almost equivalently to NNModel. Again the reason for this almost perfect emulation is the mathematical equation on which MCPLXS - H is based (see chapter 5.2).

	relative	absolute
Mean Error	0.007	0.156%
Relative max. Error	0.030	0.670%
Absolute max. Error	0.030	0.670%

Table 31: Results of MCPLXS - H with Qnet in Test 1

	relative	absolute
Mean Error	0.048	0.931%
relative max. Error	0.158	3.322%
absolute max. Error	0.158	3.322%

Table 32: Results of MCPLXS + H with Qnet in Test 1

	relative	absolute
Mean Error	0.447	9.051%
relative max. Error	1.396	31.914%
absolute max. Error	1.494	23.070%

Table 33: Results of MCPLXS - cal with Qnet in Test 1

Table 32 shows the performance of the networks for the determination of MCPLXS + H. With a relative error of 0.931% and a maximum error 3.322% Qnet performs much better than NNModel. The reason for this is probably the Learn Rate Control (see chapter 6.1.4) because this feature helps the network to converge faster, and the learning was stopped after 10,000 iterations for both software packages.

The results are not graphically displayed because the figures for MCPLXS – H as well as for MCPLXS + H would be very similar to the previous chapter except that due to a better adaptation the dots for MCPLXS + H will be located a little bit closer to the 45° line.

With an error of 9.051% and a maximum error of 31.914% the results of MCPLXS – cal are not as good as in the previous chapter (see Table 33). During a repetition of the training it was very striking that the error steadily declines at the beginning of the training but after about 3,000 learning steps the error of the test set begins to increase while the error of the training set continues its decrease.

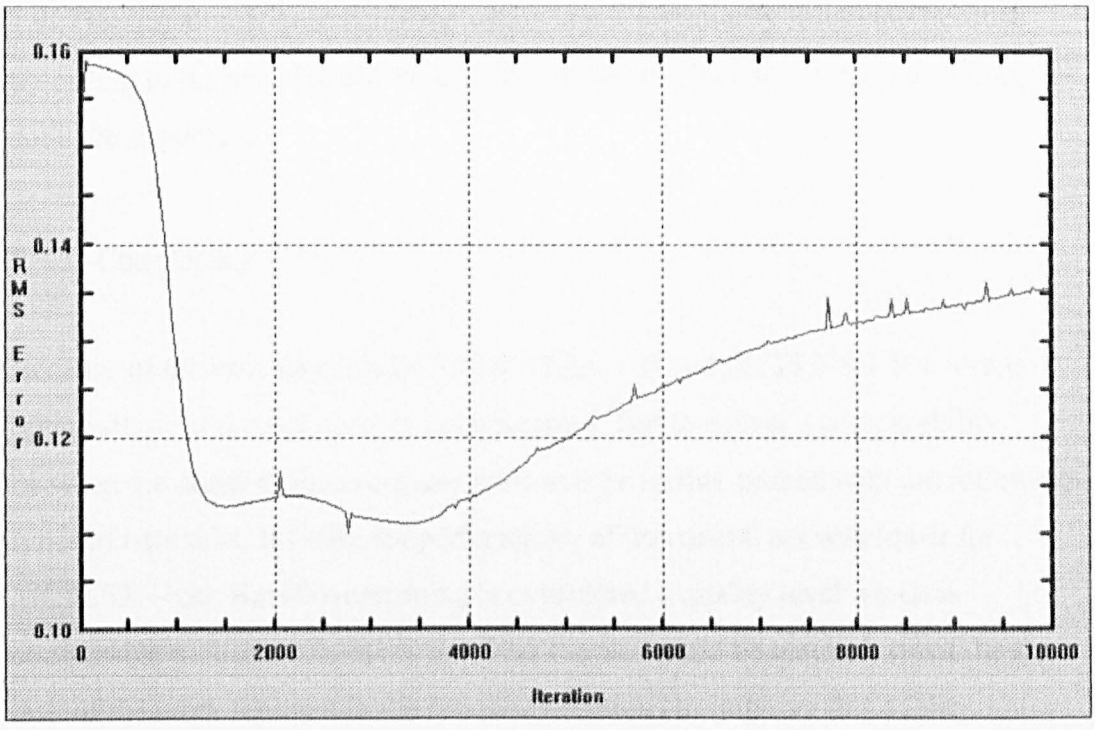


Figure 58: Error of Testing data versus learning steps

Figure 58 graphically displays the course of the Root-Mean-Square error (RMS) for the complete training of the test set.

After a rapid decrease down to about 11% a continuous increase up to about 14% can be observed. The number (14%) does not conform with the calculated mean error of 9.051% in Table 33. The reason for this is that the formula to calculate RMS in Qnet software is different from the formula that is applied in chapter 5.2. To make a comparison between the results of different software programs and tests possible, the errors are manually calculated according to the formula in chapter 5.2.

Even though the numbers are not identical it can be recognised by the course of the error graphs that overtraining took place (see chapter 6.1.5 on Learning Modes). To determine the performance of this test before overtraining occurred those weights for the neural net were chosen when the test error has reached its minimum during the training process.

With this adjustment the mean error could be further decreased by almost 1% down to 8.355%. Now the performance seen in chapter 7.4.2 (mean error of

7.927%) has almost been achieved which is not necessarily satisfying because according to the tests for MCPLXS – H and MCPLXS + H a better performance could be expected.

7.5.3 Conclusion

Because of the excellent results for MCPLXS – H and MCPLXS + H a further optimisation of the networks is not necessary. But to ensure a comparability between the chapters these complexities will be further trained with the following types of networks. Initially the performance of this neural net was lower for MCPLSX – cal. But if overtraining is considered a quality level which is comparable to the results in the previous chapter could be reached. It will be the task of the work reported in the following sections to improve this quality.

7.6 Test 2 for the Hogout problem

For the determination of the costs in Test 2 (see chapter 6.3) some adaptations of Test 1 were made. Due to the very steep course of the function and the widely spread data range some data-sets have been ignored and the outputs have been logarithmically transformed.

Since the outputs of the present data-set are not widely spread there is no need to perform an adaptation of the data as was necessary in Test 2 of chapter 6.3 and therefore this test will be skipped.

7.7 Test 3 for the Hogout problem

7.7.1 Preparation

In order to use the maximum normalisation range [0;1] in this test the data will be normalised according to chapter 6.4.1. To avoid adulteration by overtraining the neural networks are trained until a local minimum for the RMS-Error of the test

sets has been found (see chapter 6.1.5). In some cases the number of learning cycles can exceed the previous applied 10,000 learning steps.

When a local minimum was found the weights of the neural net are followed back to the stage when the local minimum has been reached in order to achieve optimal estimating conditions. This procedure is only applied for MCPLXS – cal because in most cases the error of the other complexities steadily declines during the training phase.

Overtraining could be recognised for MCPLXS – H and MCPLXS + H in only very few cases. Because the errors of these complexities are is so small these variations are not analysed further and therefore the training will be aborted after 10,000 learning steps.

7.7.2 Results

Table 34 shows the performance of the networks for the determination of MCPLXS - H. A relative mean error of 0.193% and a maximum error of 3.658% shows that this test performs less well as the previous test but the results are quite similar. An investigation of the data material revealed that data set no. 114 (see Table 26) again showed the maximum error. Due to its particularly large length (3050) which, as already mentioned, has a high influence on the output of the network, the output for this data set was considerably overestimated by the neural network. This value was already acknowledged in chapter 7.4.2 as an outlier for the determination of MCPLXS + H. If this value was removed out of the test set the relative mean error would drop to 0.116% which is less than for Test 1.

Table 35 shows the performance of the networks for the determination of MCPLXS + H. With a relative error of 0.848% and a maximum error 3.309% the results are slightly better than in Test 1. Indeed this improvement does not necessarily have to be the result of a better normalisation method. Every test set has been chosen randomly and therefore minor differences are expected. This also applies for the determination of MCPLSX – H. The statistical outlier should be mentioned here because this value was not part of the test set in Test 1.

	relative	absolute
Mean Error	0.009	0.193%
relative max. Error	0.157	3.658%
absolute max. Error	0.157	3.658%

Table 34: Results of MCPLXS - H with Qnet in Test 3

	relative	absolute
Mean Error	0.043	0.848%
relative max. Error	0.157	3.309%
absolute max. Error	0.157	3.309%

Table 35: Results of MCPLXS + H with Qnet in Test 3

	relative	absolute
Mean Error	0.408	7.970%
relative max. Error	1.009	20.031%
absolute max. Error	1.132	17.749%

Table 36: Results of MCPLXS – cal with Qnet in Test 3

Generally it can be said that for both complexities the results from the neural nets are so satisfactory that a measurable improvement of these nets can be misleading because this may derive from a different set of test results originating from different test sets.

After the elimination of overtraining MCPLSX – cal has a mean error 7.970% and a maximum error of 20.031% (see Table 36). Compared to Test 1 this result has improved by more than 1%. But again the random choice of test sets can make the small differences spurious.

7.7.3 Conclusion

No absolute statements can be made about the network when there are minor differences in the mean, relative and absolute errors still the set-up of the neural network seems to be better because two out of three nets improved the results while the third net stayed almost constant. According to the explanations of

section 6.3.2.5 this improvement could be expected because a normalisation within the maximum range [0;1] makes a distinction between the data samples easier and consequently the neural net can more precisely assign the corresponding complexity.

It is quite striking that a neural net tends to larger errors for data sets that are located in marginal segments of the normalisation range. This already has been mentioned for MCPLXS – H. But for MCPLXS + H the maximum error occurred for data set no. 13 in Table 25 because width and thickness were particularly small (normalised values = 0). Again for MCPLXS – cal the maximum error was achieved by data sample no. 114 (length = 3050).

To allow a comparison between the tests in the following chapters the training and testing data of this test will be applied.

7.8 Test 4 for the Hogout problem

In Test 4 of chapter 6 the input values were normalised in such a manner that the functional relationship between inputs and outputs becomes as simple as possible. To do so the functional relationship between each single variable and the output had to be determined by keeping all other variables constant while the input parameter that is being looked at is modified and its influence on the output is examined (see chapter 6.5.1). This procedure is only applicable if sufficient data records are present which in this example unfortunately is not the case.

Even if many samples were present this procedure still cannot be used on real world data because it can not be assumed that a functional relationship which has been determined for a special combination of other fixed input parameters will remain the same if another set of fixed input parameters is chosen. For a more detailed explanation see chapter 6.5.3. Therefore this test can not be applied in this example.

7.9 Test 5 for the Hogout problem

The aim in Test 5 of chapter 6 was not to improve the results of the previous chapter but rather to find another explanation for the relative high error of the networks. Since the data was not randomly distributed and only specific combinations of input parameters were available it became impossible for the neural net to determine a functional relationship between these parameters (see chapter 6.6.3).

In the present case insufficient samples exist to determine any patterns in the combinations of parameters. Therefore this test cannot be applied in this example.

7.10 Test 6 for the Hogout problem

7.10.1 Preparation

As in chapter 6.7 the data is trained in such a manner that the training values are spread evenly over the complete hyper space to make a distinction between different values as easy as possible. The determination of a functional relationship is replaced by the investigation of similarities between the components. Therefore the following normalisation rule is applied:

5. Determine the maximum and minimum value of a variable
6. Determine the number of different values for this variable
7. The minimum value is set to 0 and the maximum value set to 1
8. The normalised distance between each group of values is calculated as:
   normalised distance = 1/(amount of different values - 1)

7.10.2 Results

According to Table 37 the relative mean error of MCPLXS – H. is 0.170% and the maximum error is 1.256%. Again a slight improvement has been achieved compared to the previous chapter. Even though the same testing data has now

	relative	absolute
Mean Error	0.008	0.170%
relative max. Error	0.054	1.256%
absolute max. Error	0.054	1.256%

Table 37: Results of MCPLXS – H. with Qnet in Test 6

	relative	absolute
Mean Error	0.040	0.821%
relative max. Error	0.191	3.868%
absolute max. Error	0.191	3.868%

Table 38: Results of MCPLXS + H with Qnet in Test 6

	relative	absolute
Mean Error	0.368	8.281%
relative max. Error	2.187	49.880%
absolute max. Error	2.187	49.880%

Table 39: Results of MCPLXS – cal with Qnet in Test 6

been used in order to allow a better comparability between the tests, again it cannot be stated for sure that this improvement has been achieved by a better normalisation method. Since the improvement is so little and because the errors are so small these slight differences might originate from the fact that the initial weights before the training phase are chosen randomly. As a consequence the learning curves of the network are different from each other and therefore the performance of each network can again be different.

If you look at the results of MCPLXS + H in Table 38 the performance has been improved again for the mean error (0.821%). Compared to Test 3 the maximum error went up slightly to 3.868%. Since the mean error can be ascribed a higher importance than the maximum error which refers to one value only while the mean error is a characteristic of the complete database the author dares to affirm that an improvement of the networks results has been achieved.

It is striking that the data set with the maximum error for MCPLXS + H (set 13 in Table 25) is the same as in chapter 7.7 (Test 3 for the Hogout problem). The

normalised values of this data set are 0 for width and thickness because their values are the lowest in the database. Because its length is quite small too, it is not surprising that such outliers lead to extremely high errors if they are excluded from the training set and consequently included in the test set.

After the elimination of overtraining MCPLXS – cal has a mean error of 8.281% and a maximum error of 49.880% (Table 39). At first sight this result appears to be worse than the result in Test 3 for the Hogout problem in section 7.7 but the only reason for an increasing mean error is one extreme high maximum error for data set no. 1. If this value is erased from the test set the mean error drops by almost 1% to 7.356% and this test then also achieves better results than Test 3.

The following section will give a closer insight into those data sets which lead the network to extreme errors and at the same time the behaviour of neural networks will be further discussed.

7.10.3 Errors

This section will inspect those data sets which lead to particularly high errors. Test 3 and Test 6 are compared. Furthermore the error for MCPLXS – cal will be illustrated in more detail.

The most conspicuous data set is no. 114 which is responsible for the maximum error in both tests for MCPLXS – H and in Test 3 for MCPLXS – cal. The output of this data set is also among the highest errors for the other three test series namely, MCPLXS + H in Test 3 and Test 6 and MCPLXS – cal in Test 6.

As already mentioned in chapter 7.4.2 the reason for this is its exeptional length. Because in all samples the corresponding complexity was overestimated it can be assumed that according to the neural net an increase of Length of a component leads to an increased complexity. This link agrees with the classical model for the determination of the complexity according to the formula in chapter 5.2. because if Length exceeds 1 then a value for Z which depends on the actual Length is added to MCPLXS. Although PRICE partly uses different input parameters there

are still similarities regarding the influence of Length on the complexity which lead to the conclusion that the neural network behaves correctly.

A large overestimate is not surprising if you consider that the neural net has been trained with conventional lengths between 70 and 1100 and it learned to react very sensitively on any changes of the length.

The other outliers are not as striking as data set no. 114. They appear only sporadically and do not occur for all complexities in all tests. They all have in common that at least one of their normalised values is either 0 or 1 and therefore these data sets are located in the boundaries of the data and easily become outliers if they are excluded from the training set.

If you look at the contribution of inputs for MCPLXS – cal and MCPLXS + H (see Figure 56 on page 77 and Figure 54 on page 77) you cannot only see that Length significantly influences the complexity but also that the contributions of the inputs for both complexities are almost identical. There seem to be only minor differences between the equation that is applied at DASA (MCPLXS + H) and the real data.

What are these minor differences between MCPLXS + H and MCPLXS – cal?

Section 7.5.3 mentions that MCPLXS + H overestimates all complexities below 5.1 and underestimates all complexities above 5.5 (see Figure 57 on page 77). This is because MCPLXS + H behaves somewhat lethargically.

Broadly PRICE calculates the costs of a component by using a function of the product of weight and complexity:

cost = MCPLXS * weight

To specify values for MCPLXS – cal complexity values were derived from the real costs and weights of some components:

$$MCPLXS = \frac{cost}{weight}$$

For the applied samples the main part of the costs result from the milling process and therefore the total costs of a component increase as its weight decreases. An increasing enumerator (costs) and a decreasing denominator (weight) make MCPLXS grow exponentially.

Replacing the costs with the formula for the consideration of the hogout in chapter 7.3

$$MCPLXS + H = \frac{target\,OPC}{weight} = \frac{basicOPC + (basicOPC * hogout * K)^{KEXP}}{weight}$$

merely adds a correction value to the basic OPC which depends on the hogout; K and KEXP are fixed values which remain constant for identical components. The exponential increase of this function is not sufficient to keep up with the real growth of the complexity. Figure 57 on page 77 shows that the PRICE model has the tendency to underestimate particularly large complexities and vice versa.

So far in all tests the hogout of a component was represented by its weight. The raw weight can be calculated by multiplying the specified values Length, Width and Depth. The difference of raw weight and weight determines the absolute hogout and with this the relative hogout can be calculated. Therefore additional information appeared to be redundant for a neural net.

In chapter 6.5 it had been demonstrated that it is advantageous to normalise the inputs in such a manner that the functional relationship between input and output becomes as simple as possible (ideally linear) because of fewer undulations in the error mountain this relationship can be more easily determined by the backpropagation algorithm.

On the other hand the hogout plays a significant role for the determination of MCPLXS – cal and therefore the neural network should have the opportunity to set up a direct reference between absolute or relative hogout and complexity, instead of having the extra burden to recognise a relationship between the volume of a component.

In the following test it will be investigated if and to which extent a neural network can improve its performance if the relative hogout, the absolute hogout and the raw weight are added to the input values.

7.10.4 Conclusion

It can be asserted that all three types of networks of his section lead to better results than those in chapter 7.7. Therefore it can be asserted that the normalisation method applied can lead to better results and that the improvement is not attributed to a fortunate random distribution of the weights at the start of the training. Even if this normalisation method only minimally improves the results the present neural networks achieved a quality that already slightly exceeds the estimating method which is used at DASA (calculation of MCPLXS + H). While MCPLXS + H deviates from MCPLXS – cal by an average of 7.591% (see Table 30) in this test a mean error of 7.356% has been achieved.

7.11 Test 7 for the Hogout problem

7.11.1 Preparation

In order to determine if the direct input of hogout data can lead to better results in this test further columns are added to the input values. To give the neural net the opportunity to recognise a coherence between the dimensions of a component and its weight, the raw weight is added. Even though the raw weight and the weight of the component can be used to calculate the hogout, the absolute and the relative hogout are added as extra input values to the database. All records are manually normalised according to chapter 7.7.

7.11.2 Results

Table 40 shows the performance of the networks for the determination of MCPLXS - H. A relative mean error of 0.173% and a maximum error of 2.517 %

	relative	absolute
Mean Error	0.008	0.173%
relative max. Error	0.110	2.517%
absolute max. Error	0.110	2.517%

Table 40: Results of MCPLXS – H. with Qnet in Test 7

	relative	absolute
Mean Error	0.039	0.748%
relative max. Error	0.178	3.597%
absolute max. Error	0.178	3.597%

Table 41: Results of MCPLXS + H with Qnet in Test 7

	relative	absolute
Mean Error	0.351	7.003%
relative max. Error	0.835	17.815%
absolute max. Error	0.864	13.549%

Table 42: Results of MCPLXS – cal with Qnet in Test 7

shows that this test performs only 0.02% better than test 3. This result could be expected since the additional input values do not reveal any further information for the determination of MCPLXS – H because this complexity does not consider the hogout. Figure 59 supports this statement because it shows the contribution of each input variable for the output. Compared to Figure 52 in the first test in chapter 7.4 Length and Precision remain the main parameters which determine MCPLXS – H. All additional values hardly influence the networks output.

Table 41 shows the performance of the networks for the determination of MCPLXS + H. With a relative error of 0.748% and a maximum error 3.597% the results are 0.1% better than in Test 3. In Figure 60 the contribution of each input variable for the output of this test is shown. Figure 54 on page 77 of chapter 7.4 showed that the only variable that represents the hogout influenced the output of the neural net by almost 30%. In this example the hogout is represented by the raw weight, the weight of the finished component, the relative and the absolute hogout. These variables together determine the output by more than 50%.

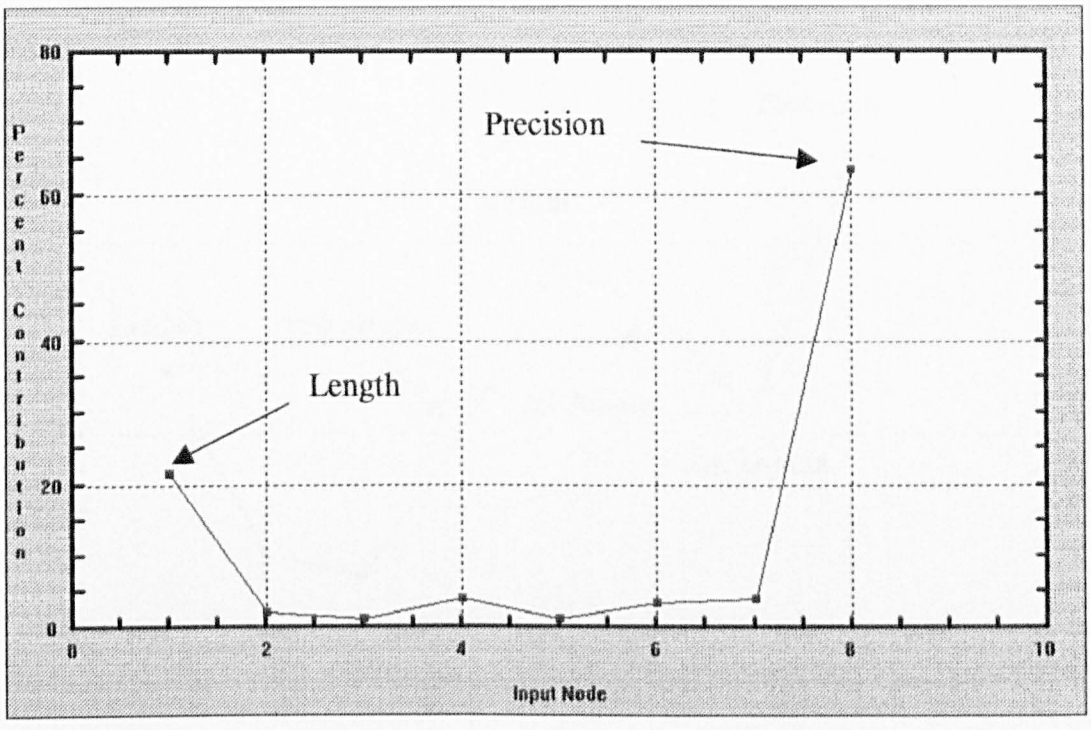


Figure 59: Percent Contribution for MCPLXS - H of inputs for Test 7

Therefore it can be assumed that this extra consideration of the hogout leads the neural network to better results.

The statistical outlier of this test is data set no. 43 (see Table 25) because it has the lowest hogout (relative and absolute) and the lowest Precision (normalised values = 0). Even though this maximum error is relatively small the extreme values of data set no. 43 lead to the highest error for the determination of MCPLXS + H.

After the elimination of overtraining MCPLSX – cal has a mean error 7.003% and a maximum error of 17.815% (see Table 42). Compared to Test 3 this result has improved by 1.276% and represents the best results of all tests for MCPLXS – cal.

The contribution of the inputs for MCPLXS – cal (see Figure 61) has significantly changed compared to Figure 56 on page 77 in chapter 7.4. The Length lost about 10% of its importance and all dimensions together (Length, Width and Depth) contribute about 50% to the determination of the complexity. In Figure 56 the contribution is about 70%. Now the hogout parameters (weight and both types of hogout) influence the neural net by almost 40% while in chapter 7.4 the

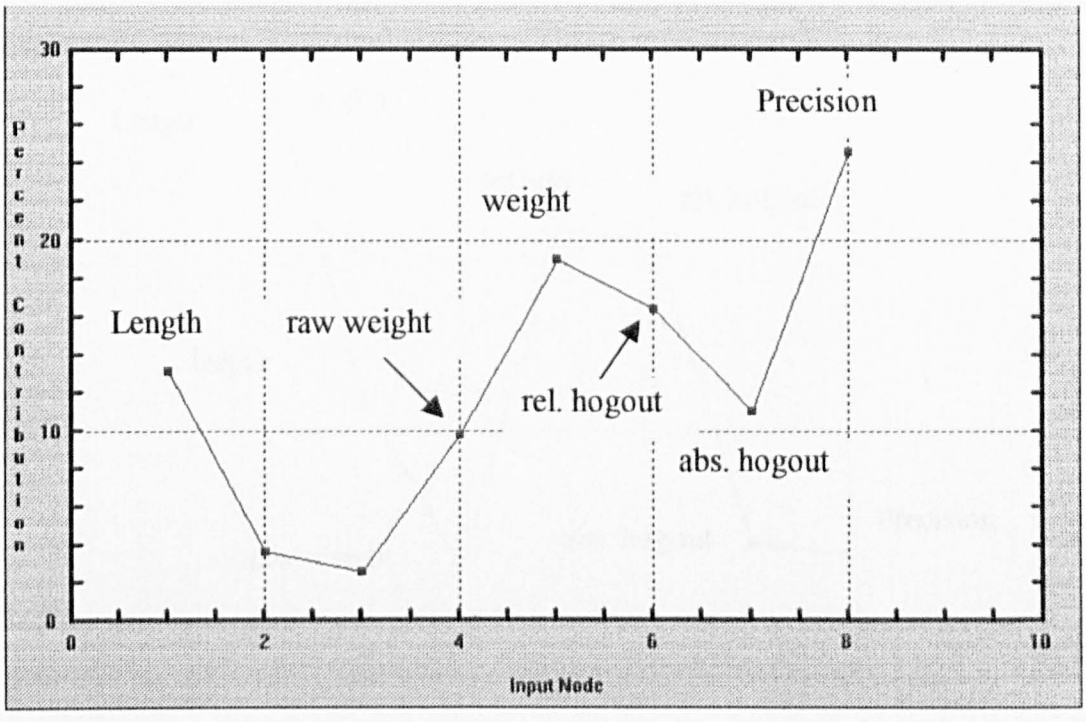


Figure 60: Percent Contribution for MCPLXS + H of inputs for Test 7

contribution comported only 15%. Therefore it can be said that the main contributions for MCPLXS – cal have been relocated towards the hogout.

7.11.3 Conclusion

At the beginning of this test it was thought that the weight of a component might not be sufficient to consider the hogout for the determination of the complexity. Although the other values (relative hogout, absolute hogout and raw weight) could be derived by Length, Width, Depth and Weight they still have been added as extra input parameters in order to make the consideration of the hogout as easy and clear as possible for the neural net.

Although the weight is still among the most important input values the addition of extra parameters seem to improve the consideration of the hogout and to meliorate the performance of the neural nets.

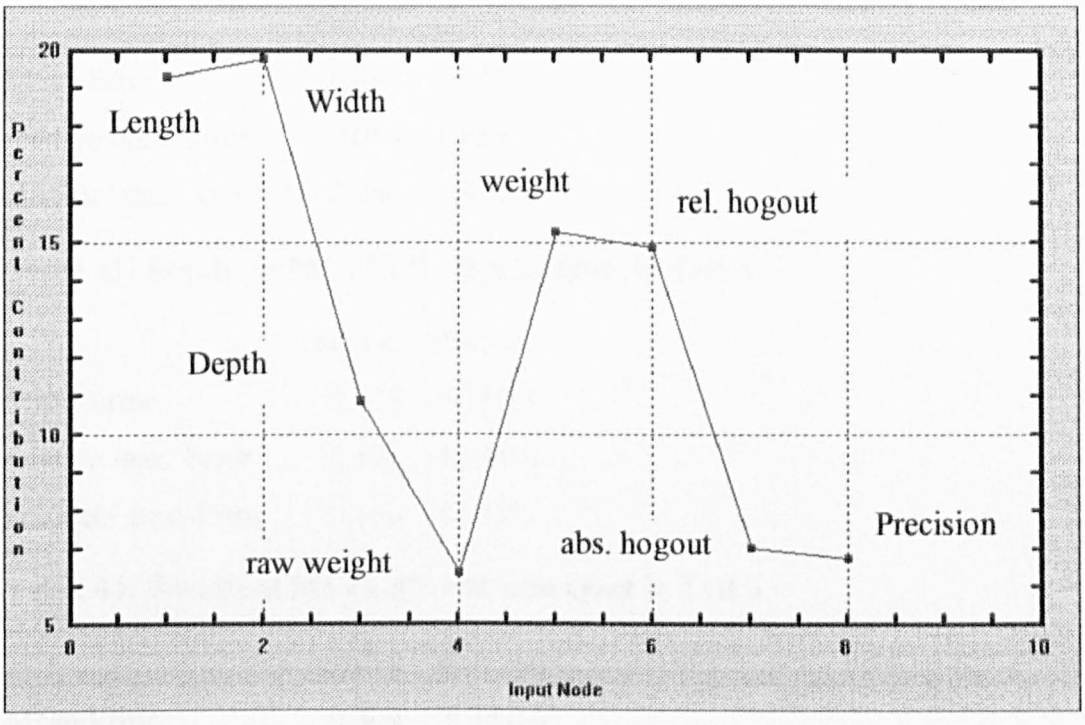


Figure 61: Percent Contribution for MCPLXS – cal of inputs for Test 7

7.12 Test 8 for the Hogout problem

7.12.1 Preparation

The set up of Test 7 is identical to Test 3 except that further input parameters have been added. Test 6 slightly improved the results of Test 3 because the inputs have been normalised in such a manner that they were evenly distributed in the data space. Now the question is whether this normalisation method can also improve the results of Test 7. Therefore the new input parameters of Test 7 are normalised according to chapter 7.10.1.

7.12.2 Results

Hardly any changes have been achieved for MCPLXS – H. and MCPLXS + H. According to Table 43 and Table 44 the relative mean error is 0.141% without hogout and 0.735% including hogout. The maximum error is 1.956% (4.133% for MCPLXS + H) and again one of the maximum errors is data set no. 114.

	relative	absolute
Mean Error	0.006	0.141%
relative max. Error	0.084	1.956%
absolute max. Error	0.084	1.956%

Table 43: Results of MCPLXS - H with Qnet in Test 8

	relative	absolute
Mean Error	0.037	0.735%
relative max. Error	0.196	4.133%
absolute max. Error	0.196	4.133%

Table 44: Results of MCPLXS + H with Qnet in Test 8

	relative	absolute
Mean Error	0.304	6.347%
relative max. Error	0.796	17.232%
absolute max. Error	0.796	17.232%

Table 45: Results of MCPLXS – cal with Qnet in Test 8

Although some improvement has been achieved it must be considered that minor differences might originate from the fact that the initial weights before the training phase were chosen randomly. With this the learning of the networks are different from each other and therefore the performance of each network can be different again. Principally no absolute statements can be made for MCPLXS – H and MCPLXS + H because their performance has been almost perfect within the last tests and because their changes were too small to come to any certain conclusions.

This normalisation method definitely achieved the best results for MCPLXS – cal because the mean error dropped down to 6.347% and the maximum error is only 17.232% (see Table 45). An examination of the contribution of each input for the output shows that compared to the previous chapter again Length and Width lost some of their importance in favour of all hogout parameters (see Figure 62).

For MCPLXS – H and MCPLXS + H the contribution of the inputs is not displayed because it has hardly changed from the previous chapter since these

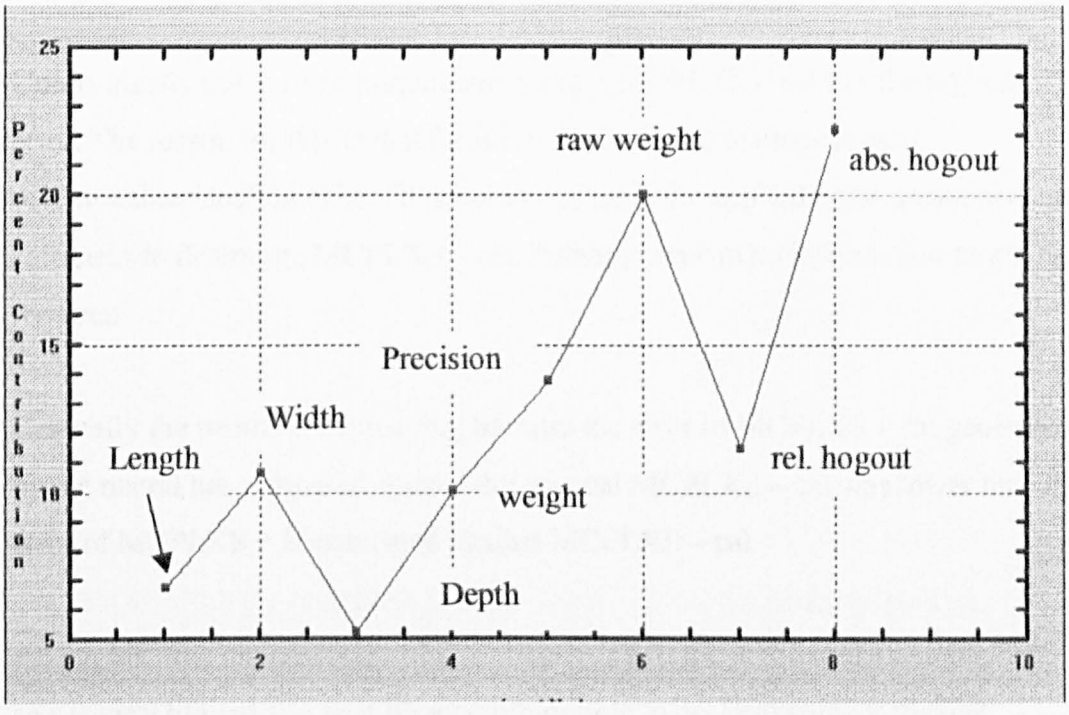


Figure 62: Percent Contribution for MCPLXS – cal of inputs for Test 8

values are mathematically determined. Slight changes occur due to the fact that the normalisiation method has changed which might have a little distorting influence on the mathematical equation.

7.12.3 Conclusion

Again with this last test MCPLXS – cal could be slightly improved. By now the predictions of the neural nets are definitely better than the PRICE model. The following section will summarise the procedures described in this chapter which led the neural network to achieve its good results.

7.13 Summary of all Tests

Although Test 8 obtained the best results the overall improvement of all tests in chapter 7 is far from being as good as those in chapter 6 (Multi-Layer Perceptron to determine Costs). In chapter 7 the results could be improved by only about 3% whereas in chapter 6 the error has been reduced by several orders of magnitude. On the other hand the initial error of this chapter was several orders lower than in chapter 6 which compensates this small progress.

Unfortunately the most important complexity, MCPLXS – cal has the highest error. The reason for this is that for this complexity no mathematically determinable function exists. It is not even sure if the applied input parameters are sufficient to determine MCPLXS – cal. Perhaps some extra information might be required.

Generally the results are satisfying because the error of MCPLXS – cal generated by the neural net compared against the original MCPLXS – cal was lower than the error of MCPLXS + H compared against MCPLXS – cal.

An aggravating circumstance of this chapter is that the amount of available data sets is very limited and so it became impossible to perform some statistical examinations which might have decreased the error of the neural nets.

The general procedure of this chapter is analogous to chapter 6. After the implementation of NNModel the data was automatically normalised by Qnet. In order to use the maximum normalisation range in the next test inputs and outputs were manually normalised according to chapter 6.4.1. In test 6 the data was normalised in such a manner that all inputs were evenly distributed in the hyper space. Already in chapter 6.7 it was quite surprising that this kind of normalisation lead to the best results because any existing functional relationships between inputs and output are very likely to be blurred applying this normalisation method. But again in this chapter excellent results have been achieved with this method although the data base and the functional relationships are completely different.

Because this normalisation method has not been mentioned in any published literature the author spoke with some experts in the field of neural networks in order to find their views.[89] [90]

---

[89] Meeting between Prof. Dr. Bernd Blümel and Gregor Sandhaus at Märkische Fachhochschule Hagen, Hagen, on 24th October 1997 at 16:00 p.m.

As well as the reasons set out in chapter 6.7 there is another reason for this good performance. The equal distribution of the data in the hyper space has the effect that the error mountain becomes generally flatter because the peaks of the mountains which are created by the clustering of data are now more widely spread. A flatter error landscape makes it easier for the backpropagation algorithm to find the minimum of the error function.

A disadvantage of this normalisation method is that if simple mathematical relationships between input and output exist they will probably be destroyed or blurred. Another restriction is that all values need to be discreet. If these values are continuous this procedure becomes much more difficult because outputs which lie between the normalised values need to be renormalised by interpolation.

In the tests of this thesis the input values were treated as discrete values (although they are continuous) because only specific combinations of inputs exist and hence this normalisation method was applied. The output on the other hand was treated as a continuous function because too many different numbers exist and consequently a "classical" normalisation method was applied.

In conclusion it can be said that this normalisation method is definitely an interesting and favourable way to handle the data of this thesis and it is probably advantageous in many other cases but it cannot generally be used on data for the above mentioned reasons. It might be interesting to evaluate whether this procedure is also applicable for other types of networks e.g. Kohonen networks or K nearest neighbours.

A closer examination of all tests that have been executed lead to the result that the hogout might not be sufficiently considered because the network would have to derive this value from the weight. In addition to that it was previously mentioned in chapter 6.5 that the functional relationships between input and output should be

---

[90] Telephone call between Stefan Mätschke at Siemens AG, Nürnberg, and Gregor Sandhaus on 5th of February 1998 at 12:30 p.m.

as simple as possible in order to obtain a flat error mountain so that the gradient descending method can quickly reach the global minimum of the error function.

Therefore an additional test was performed in which further hogout parameters were added. This test (7) was deliberately normalised according to chapter 7.7 in order to maintain the functional relationships.

This additional hogout information obviously helped the neural network in the determination of the complexity because so far the best results were achieved.

Finally this additional hogout information was also normalised according to chapter 7.10 (Test 6 for the Hogout problem) and again the error has been clearly decreased for MCPLXS – cal. It is very likely that the functional relationships between inputs and MCPLXS – cal are not that complex so that this unconventional normalisation method blurs the results because the lowest errors have been achieved with this normalisation method twice. But as already mentioned this method is probably not suitable for all kind of data.

8 Recapitulation and Discussion

8.1 Neural Networks

Chapter 1 of this thesis gives a brief and short overview about neural networks. Different types of networks have been chosen in such a manner that the complexity of structure and learning algorithm increases with each chapter. This chapter ends with the description of some of the most frequently applied networks in the field of neuro science (2.6.4 Multi-Layer Perceptron, 2.7.1 The Hopfield Model and 2.8 Self-organizing Networks).

Considerable attention has been paid to mentioning all types of networks with their corresponding characteristics which might possibly play an important role within this thesis. At the beginning of this thesis the spectrum of neuro science was already slightly larger than it has been mentioned in chapter 1. And particularly within the last years many new types of networks and learning algorithms have been developed which have been applied in a growing range of uses.

A closer view at these new developments shows that the main principles of neuro technology have not changed. Rather the networks have been significantly modified according to their conceptual formulations and tasks. The reader of this thesis might understand why so many subtypes of networks and learning algorithms exist because in some tests the significant modification of the "classical" model leads to excellent results (see Test 6 of chapter 6.7 and chapter 7.10). Later sections of this chapter will give a clearer insight into these changes.

Regarding the set up of a neural network innumerable combinations regarding the network connections, transfer functions, learning algorithms, normalisation methods etc. exist. Therefore it is a very common problem to find the most suitable set up to analyse the given database. Even though some analytical

methods have been developed – e.g. to find the optimal number of learning samples and hidden units (see chapter 6.1.2) – it is often very difficult to apply these models. For this reason many modelers rather find an appropriate neural network by trial and error and with years of experience this procedure becomes quite intuitive.

Intuition can be quite appropriate in some mathematical areas. For example, to perform a non linear regression analysis (see chapter 4.1.1) the statistician has to detect the underlying function by just looking at the data samples. Therefore intuition can be an acceptable way to find a neural network that is able to analyse the given data sufficiently.

In chapter 1 the author decided not to give a more detailed description of each type of network or to name all types and subgroups of neural networks which have been mentioned in the literature because this would be too long and would not contribute to the ideas and problems of the following chapters.

8.2 Cost estimating in project management

The main objective of this thesis is the determination of costs or complexities because they represent the bases of the PRICE model. If you investigate the competitors cost estimating software tools other output units for the expenditure of projects can be found. One of the major implemented units is the time exposure (man-month). With the help of appropriate key of payment the time exposure can lead to the costs of a project. For the software development the Lines of Code (LoC) or function points are frequently used to calculate the costs of a software project. At Daimler Benz Aerospace Airbus the production time on milling machines is frequently used to compare metal components with each other.

A general problem in the determination of costs for components is that much information is available which cannot be quantified but still have a significant influence on the costs. Essentially three information sources exist that can be used for the determination of costs: production cost, cost centre and properties of components.

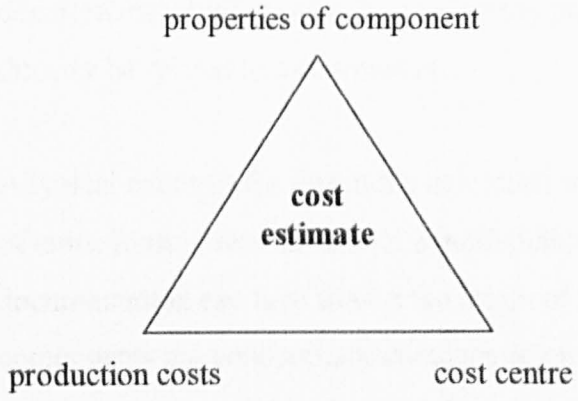


Figure 63: Cost Triangle

In Figure 63 these sources are represented in a triangle because they influence each other strongly and therefore the cost estimator should look at them as a whole.

The "properties of components" describes the general requirements that the component has to fulfil. They are mainly determined during the development or construction phase. These requirements can be subdivided for example into different levels of importance of the component. The component may be essential or it may be purely decorative. From this subdivision the technical requirements of the component can be derived and this may directly influence the costs.

In the aerospace industry for example many regulations have been developed that - depending on the components importance for air safety – specify to which extent a technical unit has to be tested and reviewed for surge immunity, tensile strength, resistance to shock etc. There are also standards for the implemented material. To ensure safety for example an international regulation is that recycled aluminium must not be used in the aircraft construction. All regulations and standardisation have to be considered during the construction phase and therefore they have a significant influence on the later production costs of a component.

The cost centre refers to all departments that are involved in the production process of the component. Not only do the average production rates of directly involved cost centres have to be considered, but also the overhead costs of

departments which service the production process as long as their input can directly be related to a component.

A typical example for this in the aeroplane industry is the detailed documentation of units. In the eventual case of a malfunction of a technical unit the documentation can help to find the origin of the defective component. For some components the concomitant documentation is so precise, that even aeroplanes more than 20 years old when, in which lot, on which machine and unit and by which labourer can be traced.

This trace allows faults on other planes where the same failure has not yet occurred to be found and further malfunctions can be prevented, thereby improving air safety. This documentation process can be very costly and although it has nothing to do with the production itself its expense should be considered in the cost estimate.

Other cost centres which have a indirect influence on the costs of a component are manufacturing control, incoming product control and quality control. For quality assurance reasons components are tested to their ultimate limits.

The production costs cover all expenses that are directly involved in the production process. Depending on the machine used and on the production process these costs can vary significantly. For this type of costs testing components and the factory rejects should be included in the estimate.

These three information sources for cost estimates can be visualised in an influence triangle. For example, if during the design phase of a component the technical requirements turn out to be extremely high then it is very likely that more cost centres will be involved in the production process, e.g. quality assurance and manufacturing control. A high technical standard and the necessity to manufacture testing components usually rises the production costs, too.

One typical problem in collecting information for the cost estimate is that one source is usually not sufficient to gather enough data and therefore other sources

need to be looked at, too. On the other hand if all sources of information are considered then a variety of classification numbers would be at the estimators disposal, i. e. technical requirements, involved cost centres, production costs. Then the question would be how these different types of classification numbers could be summarised into one result of the cost estimate.

Generally neural networks can be very suitable for this information processing. If sufficient data material is available for each source of information an individual network could be created and their results could be summarised into another neural net to retrieve the final cost estimate. One major advantage of neural networks is their flexibility regarding their implementation because they are not limited on one predefined equation. They rather represent a database and therefore any kind of summarisation is imaginable as long as it is conclusive.

8.3 The Idea

One advantage of neural networks is that any output unit can be determined by the neural network. In addition to that a combination of outputs can be generated. For example complexity and costs could be calculated at once although they are independent of each other and providing that all essential input parameters and sufficient data samples are available. For the present data material of this thesis the combination of these outputs was not proposed because during the calculation of costs in chapter 6 already considered deviations due to the recombination of the input values. If further inputs had been applied in order to calculate another output simultaneously the network would probably have been more confused.

Since neural networks are not restricted to a fixed number of inputs and outputs they can be applied for the cost estimation in almost any enterprise of almost any line of business whereas many cost estimating tools are limited to a certain business, industry or application.

Each cost estimating tool is based on its individual cost function and only with the help of some calibration tools can this function be adapted to the cost structure of an individual enterprise. PRICE, for example, uses the Maturity parameter to

calibrate the model according to a company's effectiveness. Neural networks on the other hand are not based on any predetermined function. Their estimates are simply based on the data of the company that they have been trained on beforehand. The advantage of this procedure is that no existing function has to be adapted but that the function is created according to the company's data.

With neural networks the combination of input and output parameters can be changed; more parameters can be added and non-influencing parameters can be dropped until a sufficiently precise estimate has been achieved. Other cost estimating tools are limited to their determined parameters.

The continuous addition of new data and an iterative training of the network helps the network adapt to changes of the cost structure which occur over time. It also helps to indicate that the applied parameters might not be appropriate any longer to perform a precise cost estimate and that other input parameters may have to be found.

One of the major disadvantages is the need to train the neural network with as many data samples as possible. Even a large scale enterprise like Daimler Benz Aerospace Airbus which has many excellent computer networks and much digitised information at its disposal could only obtain a very few data samples which contained real cost information. The main problem for DASA was to reduce all accessible data to a very few significant input parameters. For the selection of hogout components the following has been considered:[91]

- All components have been manufactured on similar milling-machines (single-spindle-milling-machine) in order to be based on similar production costs.
- All components are made of the same aluminium. Therefore it can be assumed that for a similar geometry the same tools (rotary grinders) and rotating and progressing speeds will be used.
- All components have been manufactured during the same period of time in order to have similar economic conditions.

[91] see chapter 7.3 Hogout Problem

- For all components identical lot sizes have been produced in order to keep the costs for tool changes constant.

More input parameters could have been used (e.g. type of milling machine, different materials, time periods and lot sizes) in order to provide more training data. On the other hand this wide spectrum of inputs leads to the necessity to use many more data samples in order to achieve a sufficient representation of the cost relationships because then many more combinations of inputs might exist which lead to the same output.

Or, to rephrase this statement: to represent a hyperplane of a two-dimensional room (i.e. straight line) at least two points (i.e. data sets) are necessary. To represent a hyperplane of a three-dimensional room (i.e. flat plane) at least three points are necessary.

If more input parameters are added the more data samples are necessary and the more dimensions the function will have, which have to be represented by the neural network. This increase of data and dimensions, rapidly slows down the learning process of the net and the required time to train the net increases exponentially. Therefore the general rule is to use only necessary input data.

Another disadvantage of the use of neural networks is that some variables are difficult to express in numerical values. For example in chapter 6.2 (Test 1) the implementation of the beginning (PSTART) and the end (PEND) was not adequate to represent the duration of the project. And it could be even more difficult to find numerical values for other input parameters (e.g. type of milling machine or different materials) that are able to develop a reasonable functional relationship between inputs and outputs. This disadvantage leads to another problem which especially affects the backpropagation learning algorithm.

During the backpropagation algorithm the error of the network is minimised with the gradient descending method. The problem of this procedure is that it is likely that only a local minimum is found and that the error remains quite high. Chapter 6.5 (Test 4) describes this problem more thoroughly. In order to avoid this

problem the data material should be prepared so that as few local minima as possible exist in the error function.

This can be achieved by normalising the data in such a manner that the functional relationship between each input and output becomes linear. A prerequisite of this normalisation is that a detailed statistical analysis has been performed with the data material to detect the functional relationships. But the execution of a statistical analysis is exactly what should be substituted by artificial intelligence.

Earlier the cost drivers have been analysed regarding their functional coherence to the costs (CER) and then a mathematical equation has been developed and this equation was used for the cost estimation in project management. The new alternative would be to determine the functional relationships between cost drivers and costs in order to obtain a linear function between input and output by normalising the data and then to train a neural network.

The effort for both models looks the same and the idea of using an artificial intelligence which autonomously detects any functional relationships for the cost estimator does not seem to be realisable. Things are not as bad as this suggests and the following paragraphs explain why.

Neural networks are definitely able to represent non linear functions because in all tests of chapter 7 outstanding results have been achieved for MCPLXS – H and MCPLXS + H and the equation on which these outputs are based on, is far from being linear (see chapter 5.3). One important reason for the very poor results of chapter 6 (Multi-Layer Perceptron to determine Costs) is that there may be steps within the function, and the distribution of the data samples on which the networks have been trained.

Chapter 7 (Multi-Layer Perceptron on real world data) proves that artificial intelligence can lead to better results than the classical cost estimation method. To consider the hogout sufficiently PRICE introduced a correction factor which tended to overestimate small complexities and to underestimate large complexities. The neural networks were able to further reduce the error without

performing complicated statistical investigations and it is very unlikely that a linear relationship between input and output exists because MCPLXS – cal is not based on any known function. Due to the lack of enough data samples these analyses could not further be executed.

8.4 The potential for future work

In this thesis only the Multi-Layer Perceptron with the Backpropagation algorithm has been used to emulate any given cost function. The choice of this type of neural net with the corresponding learning method is very obvious for this particular conceptual task. Backpropagation is mainly used for the determination of functions because it uses the gradient descending method and it is assumed that cardinal data exists and that for each input a definite output is assigned. From this population a representative sample is drawn to train the neural network. If during the testing phase an input is applied which is different from the training data then the neural network calculates an output which lies in-between those data samples. This way an output can be created for each input.

The backpropagation algorithm aims to find a continual function through the points on which it has been trained. In many tests of this thesis the error was quite high as this function did not exactly represent the data. Therefore in Test 4 (chapter 6.5) a method has been developed that departed from the basic idea of representing a continuous function. Instead each training vector is treated as a unique pattern with its individual output (i. e. costs or complexities). If during the test phase an input pattern is presented to the neural network it searches among the training patterns for those vectors that are as similar as possible to the applied input pattern and the corresponding output is generated.

This strategy describes the classification of an input because this vector is assigned to the closest corresponding values. For classification problems many neural networks exist which are specially designed for this purpose. Typical representatives are the Hopfield model (see chapter 2.7.1) and the Kohonen network (see chapter 2.8), but also other types of neural networks that have not been mentioned in this thesis have recently been recognised such as the

Boltzmann Machine, Time Delay Neural Network, Time Delay Neural Network, Counterprogation and Fuzzy Associative Memory.[92] The functional principle of all these types of networks is similar.

During the training phase the vectors are assigned to different clusters and during the test phase the neural network decides to which of these clusters the applied vector belongs and the corresponding output is generated. In many cases this vector lies in-between two or more clusters. Then the neural network often shows to which extent, in percentage terms, the vector belongs to each cluster.

To apply the classification method in the field of cost estimating in project management at first different price clusters (or complexity cluster for the PRICE model) need to be defined. A simple classification of projects in terms of cost categories as cheap, medium or expensive will most likely not be sufficient for the cost estimator. A finer subdivision is required because the estimate should be as precise as possible. Since a deviation of about ±10% between real costs and the estimate is usually accepable (see chapter 3.5.7) a subdivision of the outputs into 20%-steps may also be acceptable.

For Daimler Benz Aerospace Airbus most estimates are in the range from 5,000 to 300,000 Deutsche Marks and in this case 24 (5,000 – 6,000; 6001 – 7200; 7201 – 8,640; ...) clusters should be required. This amount of clusters can easily be handled by artificial intelligence but if it is thought that many more clusters are required it could become very difficult for a neural net to distinguish between similar clusters.

All mentioned advantages and disadvantages of this classification are hypotheses as no tests have been performed. On the other hand if you consider that a basic approach towards this direction in Test 6 in chapters 6.7 and 7.10 already lead to the best results of this thesis further research in this field looks quite productive.

---

[92] Sarle, Warren: Frequently asked questions about neural networks, URL: http://www.faqs.org/faqs/ai-faq/neural-nets/part1/, part 1 of 7: Introduction, visited on March 16th 1998

9 Summary of Abbreviations

CER	Cost Estimation Relationship
DCAA	Contract Audit Agency
IPCE	Independent Parametric Cost Estimates
L	Length of a component
M_MCPLXS	Measured Manufacturing Complexity (target value of a neural net; equals MCPLXS)
Mach or MI	Machinability (describes the difficulty in machining a material)
Mat or MA	Maturity (describes assembly difficulties due to either tight tolerances or expensive labour intensive processes)
MCPLXS	Manufacturing Complexity (generated by the PRICE-Model)
MCPLXS - cal	Calibrated Manufacturing Complexity (determined by the actual costs)
MCPLXS - H	Manufacturing Complexity where the hogout correction formula of the PRICE model is not applied (=MCPLXS)
MCPLXS + H	Manufacturing Complexity where the hogout correction formula of the PRICE model is applied
NP	Number of Parts
P_MCPLXS	Predicted Manufacturing Complexity (predicted by a neural net)
PEND	Project end date
Plat or PLTFM	Platform (establishes the specification and testing level, operating environment, and reliability requirements that the element will be designed to meet)
Prec or PRECI	Precision (governing tolerances for the fabricated parts)
PSTART	Project start date
QTY	Quantity of parts that are going to be produced
R_MCPLXS	Residual MCPLXS (=M_MCPLXS - P_MCPLXS)
WT	Weight of a component

10 Table of Figures

11 Table of Tables

12 Bibliography

- Aarts, E., Korst, J.: Simulates Annealing and Boltzmann Machines, Chichester, 1990
- Ackley, D. H., Hinton, G. E., Sejnowski, T. J.L.: A learning algorithm for Boltzmann machines, Cognitive Sciences, 9, 1985
- Augustine, Normann R.: Augustine's Laws and Major System Development Programs, in: Astronautics & Aeronautics, April 1980
- Barron, A., R.: Approximation and estimation bounds for artificial neural networks, Proceedings of the Fourth Annual Workshop on Computational Learning Theory, 1991
- Batchelder, C.A./Boren, H.E./Campbell, Jr. H.G./Dei Rossi, J.A. und Large, J.P.: An Introduction to Equipment Cost Estimating, The Rand Corporation, Memorandum RM-6103-SA, December 1969
- Daimler-Benz Aerospace Airbus - prospect: Into the future, Hamburg, Nov. 1996
- Daimler-Benz Aerospace Airbus information prospect, Hamburg, Nov. 1997
- Der Spiegel no. 14/1982
- Dowsland, K. A.: Variants of simulated annealing for practical problem solving, Applications of Modern Heuristic Methods, (ed. V. J. Rayword-Smith) Alfred Waller, Henley-on-Thames, 1995
- Durbin, R., Willshaw, D.: An analogue approach to the travelling salesman problem using an elastic net method, 1987, Nature 326:689-691
- Ebner, Clauß: Statistik für Soziologen, Pädagogen, Psychologen und Mediziner, Grundlagen, Verlag Harri Deutsch Thun, 1989
- Grün, Oskar: Beiträge zur Projektorganisation, Heft 3: Kosten und Kostenrechnung in der Projektorganisation, Wien, 1977, DBW-Depot 78-3-3 (C.E. Poeschel Verlag Stuttgart)
- H. Reschke, H. Schelle, R. Schnopp: Handbuch Projektmanagement Band 1, Verlag TÜV Rheinland Köln, 1989
- Help Manual of Qnet ver. 2.1: Neural Network Training - Learning Modes, Vesta Services, Inc. 1994, 1995

- Hinton, G. E., Rumelhart, D. E., Williams, R. J.: Learning Representations by Backpropagation Errors, Nature 323, 1986
- Hopfield, J. J., Tank, D. W.: Kollektives Rechnen mit neuronenähnlichen Schaltkreisen Spektrum der Wissenschaft, Computersysteme, 1989
- Hopfield, J.J.:Neural Networks and physical systems with emergent collective computational abilities. Proc. of the Nat. Academy of Science 79, Washington, 1982
- Jones, Ray D. and Niebisch, Klaus: Cost Estimation Techniques, INTERNET Expert Seminar on Cost Control in Project Control, Zürich, 1975
- Kinnebrock, Prof. Dr. Werner: Neuronale Netze: Grundlagen, Anwendungen, Beispiele, Oldenbourg, München, 1992
- Kohonen, T.: Self Organization and Assoziative Memory, Springer, Berlin, 1988 (2. Edition)
- Lisboa, P. G. J.: neural networks - current applications, Chapman & Hall, 1992, London, p. XVIII
- Litke, Hans-D.: Projektmangement: Methoden, Techniken, Verhaltensweisen, Hanser Verlag, München-Wien, 1993
- Madauss, Bernd J.: Handbuch Projektmanagement, Poeschel Verlag Stuttgart, 1990
- Masters, Timothy: Practical Neural Network Receipes In C++, Academic Press, San Diego, 1993
- Petters, Claus: Computer aided parametric cost estimation, DASA-LA Hamburg, 1995
- Platz, J.: Projektmanagement erfolgreich einführen, Überlegungen und praktische Erfahrungen beim Aufbau eines Einführungskonzepts, in: zfo, Zeitschrift Führung u. Organisation, 1987, Nr. 4, p.217-226
- PRICE H Reference Manual, PRICE Systems, Moorestown, New Jersey, 1988
- Rieger, Anke: Forschungsbericht Nr. 508/93, Neuronale Netzwerke, 1993, Dortmund
- Ritter, H., Martinetz, T., Schulten, K.: Neuronale Netze, Addison-Wesley, 1992

- Sarle, Warren: Frequently asked questions about neural networks, URL: http://www.faqs.org/faqs/ai-faq/neural-nets/part1/, part2, part5 and part6, last visited on March 16th 1998
- Starrett, Charles O., Jr.: Parametric Cost Estimating - An Audit Perspective, In: ISPA Journal of Parametrics, Vol. I, No. 4, Spring 1982
- Süddeutsche Zeitung from 4th of June 1982
- Vytautas Vysniauskas, Frans C. A. Groen, J. A. Kröse: The optimal number of learning samples and hidden units in function approximation with a feedforward network, Technical Report CS-93-15, University of Amsterdam, Faculty of Computer Science and Mathematics, The Netherlands, 1993
- Widrow, B. Hoff, M. E.: Adaptive switching circuits; Ire Wescon Convention Record, New York, 1960
- Zurada, Jacek M.: Introduction To Artificial Neural Systems, West Publishing Company, St. Paul, 1992