

Rapid #: -15932546

CROSS REF ID: **83598**

LENDER: **NTU :: Main Library**

BORROWER: **COP :: ArthurLakesLibrary**

TYPE: Article CC:CCL

JOURNAL TITLE: Neurocomputing

USER JOURNAL TITLE: Neurocomputing

ARTICLE TITLE: Power series and neural-net computing

ARTICLE AUTHOR: Kalveram, KTh

VOLUME: 5

ISSUE: 4

MONTH:

YEAR: 1993

PAGES: 165-174

ISSN: 0925-2312

OCLC #: 39098218

Processed by RapidX: 3/18/2020 8:36:27 PM



This material may be protected by copyright law (Title 17 U.S. Code)


16 **Rapid #: -15932546**



RapidX Upload

Status	Rapid Code	Branch Name	Start Date
New	COP	ArthurLakesLibrary	03/19/2020 03:34 AM
Pending	NTU	Main Library	03/19/2020 03:34 AM
Batch Not Printed	NTU	Main Library	03/19/2020 08:06 AM

CALL #: **QA76.87**
LOCATION: **NTU :: Main Library :: cj**

REQUEST TYPE: Article CC:CCL
JOURNAL TITLE: Neurocomputing
USER JOURNAL TITLE: Neurocomputing
NTU CATALOG TITLE: Neurocomputing
ARTICLE TITLE: Power series and neural-net computing
ARTICLE AUTHOR: Kalveram, KTh
VOLUME: 5
ISSUE: 4
MONTH: 
YEAR: 1993
PAGES: 165-174
ISSN: 0925-2312
OCLC #: 39098218
CROSS REFERENCE ID: [TN:83598][ODYSSEY:138.67.209.47/ILL]
VERIFIED:

BORROWER: **COP :: ArthurLakesLibrary**



This material may be protected by copyright law (Title 17 U.S. Code)
3/19/2020 8:06:03 AM

NEUCOM 207

Power series and neural-net computing

K.Th. Kalveram

*Abteilung für kybernetische Psychologie, Heinrich-Heine-Universität, Universitätsstrasse 1,
W-4000 Düsseldorf, Germany*

Received 29 November 1991

Revised 20 March 1992

Abstract

Kalveram, K.Th., Power series and neural-net computing, *Neurocomputing* 5 (1993) 165–174.

A power series expansion is represented by a three-layer feedforward network, the number of nodes in the hidden layer corresponding to the number of terms retained in the series, and the synaptic weights in the hidden layer representing the exponents used. The activation functions addressed to input, hidden and output nodes are log, anti-log and linear. The training rules applied to determine the weights of the output layer, that means the optimal power series coefficients, are the ordinary delta rule and a least squares based simultaneous learning rule called LSQ-rule. The performance of this network is compared to a three-layer network with the same number of nodes but sigmoidal activation functions, trained by backpropagation. With respect to the selected four functions to be approximated, the delta rule yields an output error considerably less than the error of the sigmoid network. Best results, however, are obtained combining the power network with the LSQ-rule. In this case, 'performance related to number of training trials' is – on average – about 30,000 times better, compared to backpropagation.

Keywords. Feedforward networks; power series; Taylor series; activation function; log-antilog networks; universal approximation; backpropagation.

1. Introduction

Neural networks with multilayer feedforward topology are universal approximators, realizing arbitrary continuous mappings [3, 4]. Nevertheless, they often are regarded as mysterious tools performing complex mapping and storage functions in the brain or in technical information processing systems. One of the mysteries is the number of nodes in the hidden layer(s) required for a predefined accuracy; another, whether a learning rule applied to the network forces the weights in the hidden layer(s) to arrange in distinct patterns, performing 'representations' in spaces and hyperspaces, or revealing notations like 'wolf', 'grandmother', and 'Little Red Riding Hood'. However, there exist other universal approximators, well known and frequently used in the sciences, without any hidden property: power series, for instance, or as a special case, the Taylor series. The present paper shows that such a power series can

Correspondence to: K.Th. Kalveram, Abteilung für kybernetische Psychologie, Heinrich-Heine-Universität, Universitätsstrasse 1, W-4000 Düsseldorf, Germany.

be represented by a network with three-layer feedforward topology, the number of nodes in the middle ('hidden') layer corresponding to the number of terms retained in the series, while the synaptic weights in the hidden layer represent the exponents.

2. Theory

2.1 Definition of the power network

In case of two variables x_1, x_2 the power series is given by

$$y_k(x_1, x_2) = \sum_{i,j} k C_{ij} x_1^i x_2^j, \quad k = 1, 2, \dots, L, \quad (1)$$

where L denotes the number of functions to be considered.

Assuming $L = 2$, Fig. 1 shows the network (or the circuit diagram referred to an analog-computer) which realizes (1), restrained to arguments $d \leq x_1, x_2 \leq 1 - d$ ($0 < d < 1$), and exponents $0 \leq i, j \leq 2$, each processing element only using addition to combine the inputs (abbreviation: ${}_1C_{ij} = a_{ij}$ and ${}_2C_{ij} = b_{ij}$).

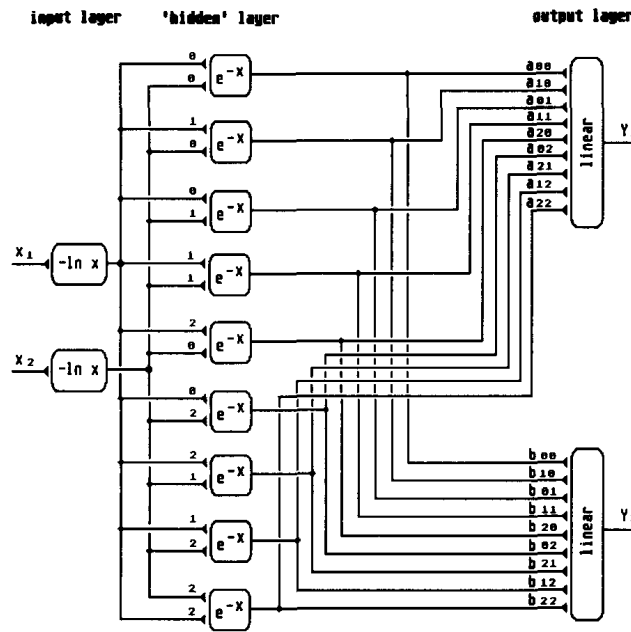


Fig. 1. Feedforward network, representing two power series expansions of order two. All terms with exponents > 2 are omitted. The weights 0, 1, 2 of the hidden layer correspond to the exponents used. The weights a_{ij}, b_{ij} ($0 \leq i, j \leq 2$) of the output layer represent the coefficients of the corresponding power series. In order to ensure positive output values of all input and hidden neurons, the input variables x_1 and x_2 are allowed to vary only between d and $1 - d$ ($0 < d < 1$).

The input layer applies the logarithmic function $-\ln(x)$ as activation function. Since the inputs vary between 0 and 1, the related outputs remain positive. The weights in the hidden

layer are 0, 1, 2 corresponding to the exponents being used. Nine hidden nodes are provided, eight are related to the different product terms connecting the input variables in (1). The ninth node (see first line in *Fig. 1*) is a 'blind' one and generates a constant output independent from input, providing the 'offset levels' a_{00} and b_{00} . The activation function used in the hidden layer is the anti-logarithm e^{-x} . Therefore, the input signals, passing through the first and second layer, become exponentiated and multiplied as indicated in (1). The activation function of the output layer is linear, and the weights are the coefficients a_{ij} , respective b_{ij} . This layer, therefore, weights and sums up the terms put out by the hidden layer, yielding y_1 respective y_2 without further transformation. Obviously, these considerations can easily be generalized to arbitrary numbers of input and output variables, and of hidden nodes.

Besides of the logarithmic transformation, the output activity of an input layer neuron can be described as a 'resting activity' ($f_0 = -\ln d > 0$), which is the more inhibited, the greater the input signal is. Regarding a hidden layer neuron, its output varies between 1 and 0, and again is inversely related to the input. Expressed in neurophysiological terms, the connection between the input and hidden layer therefore is a disinhibitory one.

In the following considerations the performance of the power network thus defined will be assessed by applying different learning rules, and by comparing it to another feedforward network frequently used in neural network research: The sigmoid network with linear activation function in the input layer, and the sigmoid function $1/(1 + e^{-x})$ in the hidden and output layer.

2.2 Training procedure

In order to assess performance, the networks under consideration are trained by supervised learning. After training, their output errors are compared. *Figure 2* indicates a part of the test procedure the networks are embedded into. In order to ensure comparability of both networks and the various training conditions (see below), the inputs and outputs need some normalization. The input normalization transformation applied in the present paper is

$$x_n = f_n(x) = (x - x_{\min})(1 - 2d)/(x_{\max} - x_{\min}) + d = A \cdot x - B$$

with $A = (1 - 2d)/(x_{\max} - x_{\min})$ and $B = A \cdot x_{\min} - d$, (2)

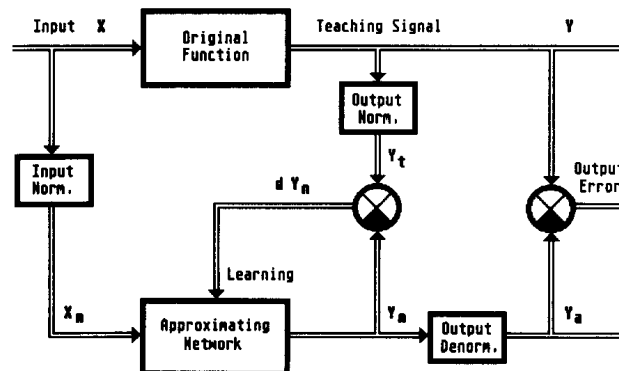


Fig. 2. Block diagram of the procedure, intended to standardize a network's training and output error measurement. The input and output normalization functions both deliver values varying between .1 and .9. Output denormalization applies the inverse of the input normalization function.

where x_{\max} and x_{\min} denote the maximal and minimal values of the variable x . (2) yields values of x_n varying between d and $1 - d$ ($0 < d < 1$). (Obviously, introducing these normalized values into (1), one gets expansions of $y_k(A_1x_1 - B_1, A_2x_2 - B_2)$, $k = 1, 2$, which may be considered as a Taylor expansion, approximating two given functions $F_1(A_1x_1, A_2x_2)$ and $F_2(A_1x_1, A_2x_2)$ at the location B_1, B_2 (see e.g. [1], p. 371, however the arguments being restricted to positive values). Regarding the power network, the input normalization prevents the outputs of the first layer from being overloaded and/or getting negative, whereas in the sigmoid network an input normalization would not be necessary, but does not affect overall operation. However, because of the output limitation of the sigmoid network to values between 0 and 1, the target values y put out by the 'teacher' must also be normalized appropriately in order to meet the requirements of the sigmoid network. The normalized teaching output is called y_t . The output normalization transformation selected in this paper is likewise (2). This transformation is applied to y also to train the power network, though this would not be necessary. Performance of the trained network then is measured computing an overall error using the difference between the original target y and the denormalized network output y_a .

In order to modify the synaptic weights of the hidden and output layer of the sigmoid network, the backpropagation rule (according to [8], pp. 324–328) is applied, with learning rate r arbitrarily fixed to 1. In the power network, only the synaptic weights w_{ki} of the output layer are assumed to be plastic, therefore the ordinary delta rule (see e.g. [8], p. 322) suffices, in this case given by

$$w_{ki} \leftarrow w_{ki} + r \cdot (y_{t,k} - y_{n,k}) \cdot z_i, \quad k = 1, 2 \text{ and } i = 1, 2, \dots, N, \quad (3)$$

where $y_{t,k}$ is the target value for the k th output node, $y_{n,k}$ the corresponding actual output of the network, z_i the output value of the i th hidden node, and N the number of hidden nodes. Learning rate r is fixed to 1/6. (This value results averaging the output weighting function $y(1 - y)$ applied in backpropagation within the limits 0 and 1).

Backpropagation and delta rule are 'sequential' learning rules based upon the 'least squares method', that means, that after a training vector $(x_n, y_t) = ((x_{n,1}, x_{n,2}), (y_{t,1}, y_{t,2}))$ has been offered to the network, the output error $dy = (dy_1, dy_2)$ is immediately fed back in order to modify the weights. The modification algorithm ensures that the squared output error related to that training trial attains a (relative) minimum. However, in the power network also 'simultaneous' learning can take place: At first a set of M training vectors is specified, thereafter the weights are computed solving a system of M linear equations, eventually by means of likewise a least squares approximation, if the system is overdetermined. For each output node ($k = 1, 2$) the equations to be solved can be written as

$$({}^j z_i) * (w_{ki}) = ({}^j y_{t,k}), \quad i = 1, \dots, N; \quad j = 1, \dots, M; \quad N \leq M, \quad (4)$$

where the superscript j indicates the running number of the training vector, and $*$ multiplication of the coefficient matrix $({}^j z_i)$ by the vector (w_{ki}) of the unknown weights. For $N = M$ the corresponding learning rule is called 'LEQ-rule', and for $N < M$ 'LSQ-rule'.

2.3 Relations to other types of networks

The power network defined above resembles the "Functional-Link net" [6]. The Functional-Link net also uses higher order terms and has three layers: the middle layer consists of the

nodes linking the input variables, each node capable of emanating a bulk of additional terms. But there exist some differences: in the power network, each node located in the middle layer generates just one term. Furthermore the Functional-Link net, “if used with some common sense, can provide rapid learning and high accuracy. However, common sense is needed to avoid an exponential increase in the number of additional ‘enhancement’ or high-order terms” ([7], p. 6). In the Functional-Link net, therefore, the functional links seem to be created, or can be omitted, using knowledge or intuition about the equations linking the input variables. In contrast, the power network automatically generates all the linkages used for approximation, without common sense being needed. Therefore, the power resp. Taylor series may explain theoretically, *why* the Functional-Link net, or other higher order neural networks, perform as well, as it has been stated by experience.

Recently, another type of feedforward network called DEFAnet (DEterministic Function Approximating network, [2]) has been proposed. In contrast to the power network, the basic idea of the DEFAnet is to subdivide the input space into rectangular cells, assuming the target function to be attained at the corners of each cell, and then to interpolate linearly the target function between adjacent grid points piece by piece. The interpolation rule within each cell simply requires of all kinds of products of input variables, however, only taken to the power 0 or 1, that is, the method interpolates linearly along the cell axes. The DEFAnet performs the multiplications within each cell likewise by log resp. antilog transformations, in this instance therefore the DEFAnet resembles the power network. However, the power network is intended to approximate a given function as a whole by a higher order polynomial approach, not piece by piece using linear interpolation. Furthermore, in the layers concerned with multiplication the DEFAnet exclusively uses excitatory synapses, whereas in the power network these synaptic connections are inhibitory.

3. Simulation

In order to test the network shown in *Fig. 1* under various learning algorithms, and to compare it to the sigmoid network, four mapping functions are selected called *P-map*, *E-map*, *FK-map* and *IK-map*:

$$\begin{aligned} P\text{-map: } y_1 &= x_1^2 - x_2^2 \\ y_2 &= x_1^2 + x_2^2 \\ (0 \leq x_1 \leq 1 \text{ and } 0 \leq x_2 \leq 1) \end{aligned} \quad (5a)$$

$$\begin{aligned} E\text{-map: } y_1 &= e^{(x_1+x_2)} \\ y_2 &= e^{(x_1-x_2)} \\ (-.5 \leq x_1 \leq 1 \text{ and } -1 \leq x_2 \leq .5) \end{aligned} \quad (5b)$$

$$\begin{aligned} FK\text{-map: } y_1 &= \sin(x_1) + \sin(x_1 + x_2) \\ y_2 &= \cos(x_1) + \cos(x_1 + x_2) \\ (-1.09 \leq x_1 \leq .623 \text{ and } .809 \leq x_2 \leq 2.98) \end{aligned} \quad (5c)$$

$$\begin{aligned}
IK\text{-map: } y_1 &= 2 \arccos(x_1^2 + x_2^2) \\
y_2 &= \arctan(x_1/x_2) - .5y_1 \\
(.057 \leq x_1 \leq .65 \text{ and } .057 \leq x_2 \leq .65) .
\end{aligned} \tag{5d}$$

(*FK*-map refers to a two jointed arm's forward kinematics, and *IK*-map to its inverse kinematics.)

Firstly, a 7×7 training grid was constructed selecting the limiting values, and 5 points equally spaced between these limits, of each input variable. Regarding sequential training, in each training trial one of the 49 gridpoints was selected at random, normalized and offered to the network's input. The normalization parameter d was fixed upon 0.1. According to *Fig. 2*, the output error was computed and used to modify the synaptic weights. A training run consisted of 16,000 training trials. In order to get statistical information, each run was repeated 10 times. At the begin of each run, the weights were initialized inserting random numbers equally distributed between -0.5 and 0.5 , however, the power network started with weights in the hidden layer fixed upon 0, 1, 2 (9 hidden nodes) or 0, 1, 2, 3 (16 hidden nodes). Regarding simultaneous training, the LSQ rule was applied to the power network, offering the 49 points of the 7×7 training grid once. The LEQ-rule only required of a 3×3 training grid (9 hidden nodes) resp. a 4×4 training grid (16 hidden nodes). The solution of the system of linear equations (if overdetermined, by means of least squares) was adopted from Späth ([9], pp. 25–32).

After training, performance was tested, using a 9×9 test grid as input pattern, constructed similar to the training grids (see *Fig. 4*). The relative output error was defined as

$$RE = \frac{(1/G) \cdot \sum_{i=1}^G y_{1,i} - y_{a1,i}}{y_{1,\max} - y_{1,\min}} + \frac{(1/G) \cdot \sum_{i=1}^G y_{2,i} - y_{a2,i}}{y_{2,\max} - y_{2,\min}}, \tag{6}$$

where $y_{k,i}$ denote the target values, $y_{ak,i}$ the denormalized actual values put out by the trained networks, $y_{k,\max} - y_{k,\min}$ ($k = 1, 2$) the range covered by the target values in each dimension, and G the number of points used in the test grid.

4. Results

Table 1 gives an overview over the main results of the simulation experiment. Most surprising are the low output error values applying simultaneous training to the power network. The power network using the LEQ or LSQ rule is also the only one which rediscovers the true structure of the *P*-map, resulting in an error exactly equal to zero. Regarding sequential training, the output errors are considerably greater, and deviate randomly from the mean from run to run, nevertheless they seem to converge. In order to achieve a better assessment of the convergence behavior, in *Fig. 3* the mean output error within training is computed every 1000 trials, results related to the four different mapping functions are averaged. Considering backpropagation and delta rule, the networks were pre-trained (99 trials) in order to get a distinct output error to start with. These errors are referred to the abscissa value '0'. The errors resulting from training by LEQ resp. LSQ rule are referred to the same abscissa value. Obviously, it is unlikely that a continued training beyond 16,000 trials will substantially

Table 1. Relative output error (RE) of sigmoid and power network, applying different learning rules, and using different mapping functions. Regarding backpropagation and delta rule, mean and standard deviation (in brackets) within 10 training runs are shown, each run enclosing 16,000 trials. Regarding LEQ resp. LSQ rule, the number of training trials only needs to equal, resp. to exceed, the number of hidden nodes.

Network	Training trials	Nodes	Mapping function							
			<i>P</i>		<i>E</i>		<i>FK</i>		<i>TK</i>	
Sigmoid	Backprop. rule 16,000	9	.109	(.005)	0.69	(.005)	.104	(.008)	.077	(.008)
		16	.108	(.005)	.078	(.011)	.092	(.002)	.086	(.013)
Power	Delta rule 16,000	9	.012	(.003)	.019	(.006)	.057	(.005)	.030	(.004)
		16	.011	(.004)	.012	(.004)	.034	(.003)	.026	(.003)
Power	LEQ rule 9/16	9	.000		.012		.030		.016	
		16	.000		.001		.005		.005	
Power	LSQ rule 49	9	.000		.009		.025		.015	
		16	.000		.001		.003		.005	

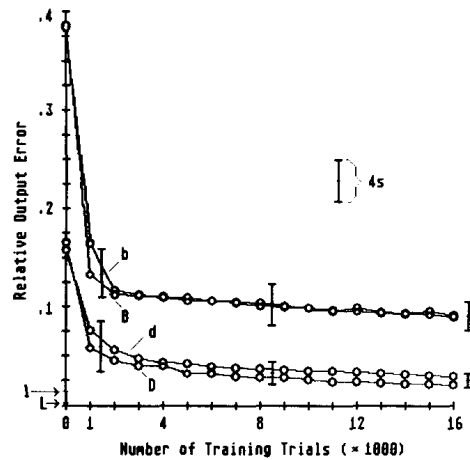


Fig. 3. Mean relative output error, related to the number of training trials within a training run. Means and standard deviations (*s*) referred to different mappings are averaged. *b, B*: Sigmoid network with 9 resp. 16 hidden nodes, trained by backpropagation rule. *d, D*: Power network with 9 resp. 16 hidden nodes, trained by delta rule. *l, L*: Power network with 9 resp. 16 hidden nodes, trained applying the LEQ rule once, using 9 resp. 16 training trials.

diminish the output error. Not shown in *Fig. 3* but worth to be mentioned is, that increased learning rates (see (3)) will enlarge the rate of convergence at the beginning of a simulation run, but as well the standard deviation at the end of the run, whereas the level of convergence is only scarcely affected. Too great a learning rate induces oscillations, therefore the learning rates in this paper are chosen to avoid this. It should further be mentioned, that the weights

in the hidden and output layer of the sigmoid network, resp. of the output layer in the power network, do not converge to the same values after different training runs, not even the signs are constant. Averaging along the weights of different runs therefore does not make any sense.

Figure 4 visualizes the maps resulting from sequential training runs exhibiting minimal output error, and contrasts it to the maps yielded by simultaneous training methods. Though only using a few trials, the latter enable the power network even to discover details of the original maps, which are totally missed by the sigmoid network.

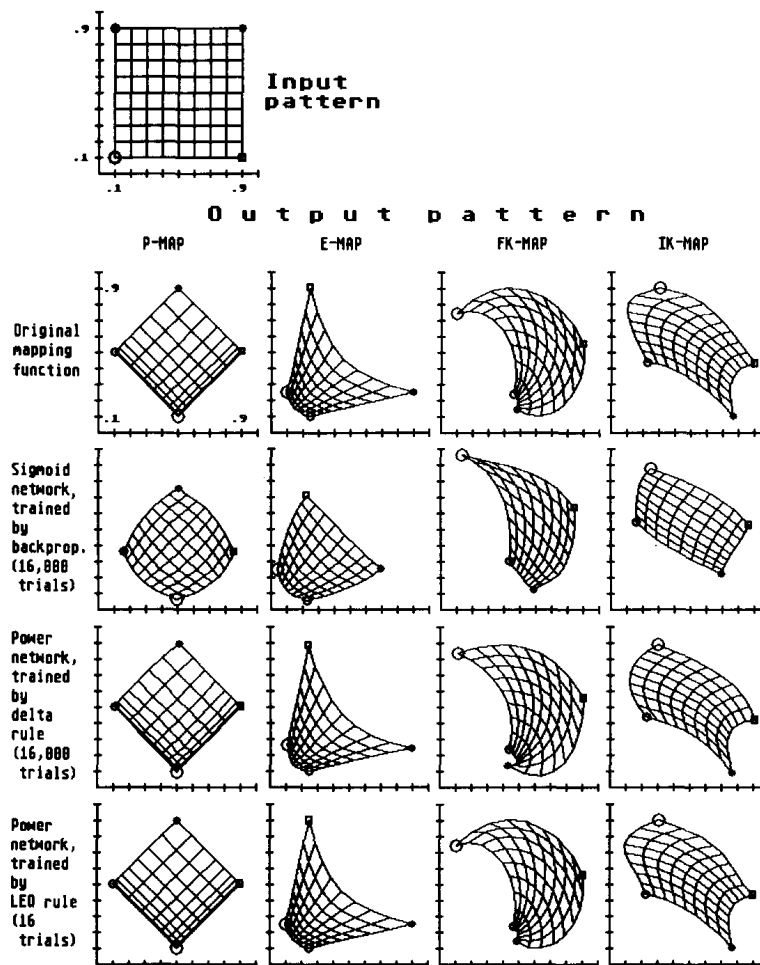


Fig. 4. Approximation of four mapping functions by the sigmoid and power network (16 hidden nodes) applying different learning rules. The examples concerning backpropagation and delta rule are taken from training runs exhibiting minimal relative output error. All mappings are shown in normalized form.

5. Implications

The paper suggests viewing a power series expansion as a feedforward neural network and proposes to use supervised learning to determine optimal power series coefficients in function

approximation problems. Referring to the selected approximation examples, best results are obtained by the power network, especially when trained using a simultaneous learning rule (LEQ or LSQ rule), compared to a backpropagation network with sigmoidal activation functions and the same number of hidden nodes. Referring to 'performance related to costs' (in this instant: the reciprocal value of the relative output error, divided by the number of training trials), the power network combined with the LEQ rule is – on average – about 33000 times better than the sigmoid network trained by backpropagation. Regarding technical applications, there is no reason to compute the product terms in (1) by log/antilog transformation, or to restrict any variable to values > 0 and/or < 1 . Removing these constraints, precision of approximation can further be enhanced. However, computational difficulties may occur when approximating a multiple input function, because of the great number of hidden nodes required even if the number of input variables is moderate. In special cases, however, very small networks may suffice to solve for seemingly complicated problems. For instance, a power network with 10 input and 6 hidden nodes is capable of precisely inverting the dynamics of an arm with two segments, only demanding one short test movement to identify all the parameters needed [5].

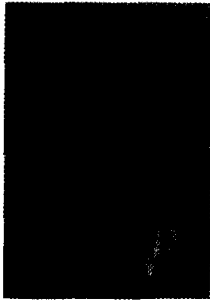
Surely, the results presented in this paper can depend on the examples chosen. But I believe that similar results will be found also in other cases, which, however, must be subject of further research.

Acknowledgement

I thank Dirk Riemann for intense and helpful discussions of the problem.

References

- [1] I.N. Bronstein and K.A. Semedjajew, *Taschenbuch der Mathematik*, 23. Auflage (Harri Deutsch, Thun und Frankfurt/Main, 1987).
- [2] W.J. Daunicht, M. Lades, H. Werntges and R. Eckmiller, Inverse kinematic with obstacle avoidance implemented as a DEFAnet, in: R. Eckmiller, G. Hartmann and G. Hauske, eds., *Parallel Processing in Neural Systems and Computers* (North-Holland, Amsterdam, 1990).
- [3] K. Hornik, M. Stinchcombe and H. White, Multilayer feedforward networks are universal approximators, *Neural Networks* 2 (1989) 359–366.
- [4] K. Funahashi, On the approximate realization of continuous mappings by neural networks, *Neural Networks* 2 (1989) 183–192.
- [5] K.Th. Kalveram, A neural network model rapidly learning gains and gating of reflexes necessary to adapt to an arm's dynamics, *Biol. Cybernet* 68 (1992) 183–191.
- [6] Y.H. Pao, *Adaptive Pattern Recognition and Neural Network* (Addison-Wesley, Reading, MA, 1989).
- [7] Y.H. Pao, Applications of neural-net computing, *Neurocomputing* 1 (1989) 4–21.
- [8] D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart and J.L. McClelland, eds., *Parallel Distributed Processing*, Vol. 1 (MIT Press, Cambridge, MA, 1986).
- [9] H. Späth, *Algorithmen für multivariable Ausgleichsmodelle* (Oldenburg Verlag, München, 1974).



Karl Th. Kalveram studied at the Universities of Bonn and Marburg/Lahn. After having completed physics and mathematics with the degree 'Staatsexamen', he studied psychology and obtained the doctoral degree. He was Professor of Psychology at the Philipps-University of Marburg, and Dean of the Faculty of Psychology. Being Professor of Psychology at the Heinrich-Heine-University since 1980, he now is the leader of the section 'Cybernetical Psychology and Psychobiology'. Here the main topics of current research are sensorimotor control of limb/tool movements, voice and articular control in fluent speakers, stutterers and aphasic patients, control of expressive mimic movements, psycho-somatic stress reactions, and modelling of individual-environmental interactions.