Neural Networks, Vol.6, pp.177-187, (1993)

# The error-bounded Descriptional Complexity of Approximation Networks

Rüdiger W. Brause,

University of Frankfurt/Main, Germany
brause@cs.uni-frankfurt.de

**Abstract** It is well known that artificial neural nets can be used as approximators of any continuous functions to any desired degree and therefore be used e.g. in high-speed, real-time process control. Nevertheless, for a given application and a given network architecture the non-trivial task remains to determine the necessary number of neurons and the necessary accuracy (number of bits) per weight for a satisfactory operation which are critical issues in VLSI and computer implementations of non-trivial tasks. In this paper the accuracy of the weights and the number of neurons are seen as general system parameters which determine the maximal approximation error by the absolute amount and the relative distribution of information contained in the network. We define as the *error-bounded network descriptional complexity* the minimal number of bits for a class of approximation networks which show a certain approximation error and achieve the conditions for this goal by the new principle of *optimal information distribution*. For two examples, a simple linear approximation of a non-linear, quadratic function and a non-linear approximation of the inverse kinematic transformation used in robot manipulator control, the principle of optimal information distribution gives the the optimal number of neurons and the resolutions of the variables, i.e. the minimal amount of storage for the neural net.

**Keywords:** Kolmogorov complexity, ε-Entropy, rate-distortion theory, approximation networks, information distribution, weight resolutions, Kohonen mapping, robot control.

**Symbols used:**

| | |
|---|---|
| $\mathbf{x}$ | input data vector to the net |
| $\hat{f}(\mathbf{x})$ | actual output of the net |
| $f(\mathbf{x})$ | desired output |
| $\delta_f$ | the maximal approximation error in a compact interval |
| $\delta^{lin}$ | the maximal error due to a linear approximation |
| $\delta^{res}$ | the maximal error due to the finite number of bits per weight |
| $V_s$ | value range of parameter *s* |
| $I_s$ | resolution (number of bits) of parameter *s* |
| $I_{sys}$ | total information (number of bits) to represent the net |
| $K_{\hat{f}}(f \mid \mathbf{x})$ | descriptional complexity |
| $K_{\hat{f},\varepsilon}(f \mid \mathbf{x})$ | error-bounded descriptional complexity |
| m | number of neurons in the net |
| n | number of neurons per dimension |
| $w_i$ | weight *i* of the first layer |
| $t_i$ | threshold *i* of the first layer |
| $W_j$ | weight *j* of the second layer |
| T | threshold of the second layer neuron |
| $y_i$ | output of neuron *i* of first layer |
| $z_i$ | activation of neuron *i* of first layer |
| S(.) | output function |
| $c_i$ | general network parameter |
| L(.) | function of the approximation error with Lagrange multipliers |

# 1 Introduction

One of the most common tasks of artificial neural nets is the approximation of a given function by the superposition of several functions of single neurons. This is especially useful for real-time, high-speed controller for industrial process control which are often implemented with descrete electronic components.

Similar to the well-known theorem of Stone-Weierstraß Hornik, Stinchcomb and White, 1989 have shown (see also e.g. Giroso and Poggio, 1990 for the property of "best approximation" function and regularization networks) that in a compact interval every function can be arbitrarily closely approximated in the $L_{\infty}$-Norm by a two layer neural network (see figure 1) when a sufficiently large number *m* of units is provided and each unit output function S(.) satisfy the conditions $S(-\infty)=0$, $S(\infty)=1$.

**Fig. 1** A two-layer universal approximation network

*Sufficiently large* - What does this mean? How do we select the appropriate number of neuronal processors for a certain application and implementation ?

Let us consider only the case of a one-dimensional output approximation, as it was done in the paper of Hornik et al., 1990. Analogous results hold for multi-output networks, i.e. vector-valued functions.

## 1.1 Error-bounded descriptional complexity

An important example for a feed-forward network is an approximation network. Let us regard an approximation $\hat{f}$ of the function f: $IR^n \rightarrow IR$ in a compact interval $C \subset IR^n$; not necessarily the best possible approximation function. For example, this can be done by the two-layer neural network of figure 1. Let the maximal absolute error of this approximation be $\delta_f$ with

$$\delta_f = \max_{\mathbf{x}\in C} |f(\mathbf{x}) - \hat{f}(\mathbf{x})| \qquad (1.1)$$

for a given approximation function $\hat{f}$.

We can regard the approximation error as a kind of discretization error. Denoting the complete value range of f with

$$V_f = |f_{max} - f_{min}|, \qquad f_{max} = \max_{\mathbf{x}\in C} f(\mathbf{x}), \quad f_{min} = \min_{\mathbf{x}\in C} f(\mathbf{x})$$

we can conclude that there are only $V_f/d$ distinguishable, fixed states of the variable f which differ by an increment of $d=2\delta_f$. All other states are undistinguishable from deviations of the fixed states. Thus, since we do not know the input distribution of $\{\mathbf{x}\}$ and therefore not the error distribution, the output has minimal

$$I_{out} = \log_2 (V_f / d) \qquad (1.2)$$

bits of information.

In the neural network network, the approximation $\hat{f}(\mathbf{x})$ depends also on the set *w* of all data bits (information) of the weights $\{\mathbf{w}\}$ of all neurons, denoted by $\hat{f}(\mathbf{x},w)$. The system parameters which determine the error of the approximation, are on the one hand the resolution of the weights or its information content

$$I_w = \log_2 (V_w/d_w) \qquad \text{with the weight increment } d_w \qquad (1.3)$$

and on the other hand the number *m* of neurons.

Certainly, when we increase the number of neurons and the number of bits per neuron the approximation will become better and the error will decrease. Nevertheless, for a certain system with a finite amount of information storage capacity (such as a digital computer) the network description information (system state) will be limited. For constant system information neither one neuron with high-resolution

weights nor many neurons with one bit weights will give the optimal answer; the solution is in between the range, cf. figure 3.5.

Therefore, we have to solve the problem: what is the best information distribution, i.e. what is the best choice for the parameters *m* and $I_w$ to maximize the Information $I_{out}$ or to minimize the approximation error $\delta_f$, using a fixed amount of system information $I_{sys}$ ?

If we regard the approximation network as a channel, we can formulate the whole problem as the task for the maximization of the *transinformation* between input and output, i.e. the determination of the channel capacity. This was done in Brause (1991). Now, let us take a different, also interesting road to the solution of the problem.

The system information $I_{sys}$ is just the number of bits we use for the representation of the weights of each neuron. Since the neural algorithm $\hat{f}$ (the network architecture) remains the same for different weight resolutions and different number of neurons, the minimization of the system information is identical to the minimization of the data size of the weights used in the network, apart from an additive constant. We can think of the neural network function $\hat{f}$ as a kind of interpreter or decoding function of the weights {**w**} on the condition that an input object **x** is given. The *descriptional complexity* (see e.g. Li and Vitanyi,1990) $K_{\hat{f}}(f|\mathbf{x})$ of the object f (the wanted output f(**x**) of the approximation network) with respect to $\hat{f}$, conditional to **x**, can be defined by

$$K_{\hat{f}}(f|\mathbf{x}) = \min \{|w|: w \in \{0,1\}^* \text{ and } \hat{f}(w,\mathbf{x})=f(\mathbf{x})\} \quad \textit{descriptional complexity} \quad (1.4)$$

and $K_{\hat{f}}(f|\mathbf{x})$ becomes infinity if there are no such *w*. The set of weight bits *w* of the network containing overall |*w*| bits can be seen as the necessary information on which the output $\hat{f}(w,\mathbf{x})$ is based: Different information *w* will result in different approximations. In contrast to computer programs which produce binary strings f on the input of binary strings **x**, the neural program is not able to approximate the wanted

output f(**x**) always exactly - generally there is a finite error depending on the number of bits for the weights used. Thus, we can define the *error-bounded descriptional complexity* $K_{\hat{f},\varepsilon}$ by

$$K_{\hat{f},\varepsilon}(f\,|\mathbf{x}) = \min\{|w|: w\in\{0,1\}^{*} \text{ and } |\hat{f}(w,\mathbf{x})-f(\mathbf{x})|<\varepsilon\} \qquad (1.5)$$

where the minimum is taken again over the sum of the number of bits of all the weights in the network at all possible assignment of bits to the weights. For the whole interval, the number

$$|w| = \max_{\mathbf{x}\in C} K_{\hat{f},\varepsilon}(f\,|\mathbf{x}) \qquad (1.6)$$

is the minimal number of bits in the network necessary to guarantee a maximal approximation error of $\delta<\varepsilon$ for the whole input interval. Our main task of computing the descriptional complexity for a concrete neural network consists of computing just this number: the minimal amount of information to describe the state of the network.

The basic idea behind this is not new. The problem of encoding an information source with the minimal number of bits without exceeding a certain error or fidelity criterion was first introduced by Shannon and Weaver, 1949 and is known as the *rate-distortion problem*, see e.g. Gallager, 1968.

Let us now consider another connection to a neighbour research field. Each number of bits for the weights in the network architecture $\hat{f}$ results in a different approximation function $\hat{f}(w)$. For a fixed number $|w|$ of bits only a fixed number of functions $\hat{f}(w)$ exists. This number is the number of possible "neural programs" and, for a certain distribution of the bits to the weights, is equal to the number of possible states of the set $w$ of all bits. If we further restrict the class $\{\hat{f}(w)\}$ by a certain error constraint, the logarithm to base 2 of the number $N_f$ of such functions is the number $|w|$ of bits:

$$H_{f,\varepsilon} = \log_2 N_f \qquad (1.7)$$

Therefore, our problem of error-constraint minimization of $I_{sys}$ becomes the problem of the minimization of the number of elements in the ε-cover of the functional class. The logarithm to base 2 of this number was termed "ε-Entropy" by Kolmogorov and Tihomirov, 1961. For neural networks, there does not exist much literature on this subject. For binary networks Williamson, 1991 computed some lower and upper limits of the ε-Entropy; the determination of the ε-Entropy for a feed-forward neural network is still missing.

In this paper, we do not only determine $I_{sys}$, the minimal number of bits for a given maximal approximation error, for a fixed assignment of bits to weights as it is necessary to determine the ε-Entropy, but we also change the assignment in order to minimize further the approximation error by the means of the principle of optimal information distribution.

## 2 Optimal information distribution

As we know, the task of computing the error-bounded description complexity for approximation networks, i.e. the system information $I_{sys}$ when a certain error is fixed, is equivalent to the task of computing the minimal error when a certain $I_{sys}$ is given. When we have different weights $w_i$, $W_j$, $t_i$, T, ... with different resolutions (number of bits per weight) in the network, the question becomes : *what is the best choice for the parameters* m *and* $I_w$, ... *to minimize the approximation error* $\delta_f$, using a fixed amount of system information $I_{sys}$ ?

The solution to this question is provided by the approach of an optimal information distribution of the neural network parameters. For this purpose let us denote the parameters m, $I_w$, ... as general system parameters $c_1$, ..., $c_k$.

## 2.1 The principle of optimal information distribution

Let us first derive the conditions for the optimal system parameters by some plausible considerations, presented in Brause, 1989, which should give a feeling for the subject and an insight into the mechanism involved. The rigorous, conventional mathematical approach will be covered by the section 2.2 afterwards.

Assume on the one hand that we transfer a fixed, small amount of information from one parameter to another (e.g. more neurons and less bits per weight) and we will find the approximation error decreased. In this case the information distribution induced by the parameter values of $c_1$, ..., $c_k$ was not optimal; the new one is better. Let us assume that on the other hand we find that the error $\delta_f$ has increased, then the information distribution is not optimal, too; by making the inverse transfer we can also decrease $\delta_f$. All subsequent changes in a non-optimal information distribution will further reduce the error until we reach a minimum. Thus, in a restricted system we have at least one local minimum of error. This extremum can be characterized by the following principle:

> In an *optimal information distribution* a small (virtual) change in the distribution (a change in $c_1$, ..., $c_k$) neither increases nor decreases the performance error $\delta_f$.

A small increment of additional information $\Delta I_{sys}$ in the system will produce a change $\Delta\delta_f$ in the maximal output error

$$\Delta\delta_f = \Delta I_{sys} \frac{\partial}{\partial I_{sys}} \delta_f = \Delta I_{sys} \sum_{i=1}^{k} \frac{\partial}{\partial c_i} \delta_f(c_1, ..., c_k) \frac{\partial c_i}{\partial I_{sys}} \qquad (2.1)$$

Each term in the sum of equation (2.4) represents an information contribution of a system parameter when we increase the overall system information $I_{sys}$. According to the principle above, an optimal distribution is given when all terms in the sum i.e. all information contributions of the system parameters are equal.

$$\frac{\partial \delta_f}{\partial c_1}\frac{\partial c_1}{\partial I_{sys}} = \ ... \ = \frac{\partial \delta_f}{\partial c_k}\frac{\partial c_k}{\partial I_{sys}} \qquad (2.2)$$

The *k* independent terms gives us (k-1) equations for *k* variables $c_1$, ..., $c_k$, leaving us with a degree of freedom of one. So, choosing the amount of available information storage $I_{sys}(c_1, ...,c_k)=I_0$, the parameters $c_1$, ..., $c_k$ are fixed and the smallest error $\delta_f$ for the particular application will result. On the other hand, for a certain maximal error a certain amount of network information is necessary.

## 2.2 Optimal system parameters

Now we want to compare the principle above to a more conventional mathematical approach.

The maximal error $\delta_f$ is a multivariate function $\delta_f(c_1,...,c_k)$. We will look for the minimal error of the system using only a certain amount of system information and search an optimal parameter tuple $(c_1^*,...,c_k^*)$ such that

$$\delta_f(c_1^*,...,c_k^*) = \min_{c_1,...,c_k} \delta_f(c_1,...,c_k) \qquad (2.3)$$

which is accompanied by the restriction that the whole information $I_{sys}$ in the system should not be changed during the maximization process

$$I_{sys}(c_1,...,c_k) = I_0 = \text{const} \qquad (2.4)$$

By these two conditions the relative minimum (2.3) of the multivariate function $\delta_f$ is searched under the restriction of (2.4). The standard method to get the local extrema of a constrained function is the method of Lagrange multipliers. For this purpose let us define the differentiable functions

$$L(c_1,...,c_k,\lambda) := \delta_f(c_1,...,c_k) + \lambda I(c_1,...,c_k) \qquad (2.5)$$

$$\text{with} \qquad I(c_1,...,c_k) := I_{sys}(c_1,...,c_k) - I_0 = 0$$

Since the function *L* includes the restrictions, the necessary conditions for a relative extremum of the function gives us the necessary conditions for optimal values of the system parameters

$$\frac{\partial}{\partial c_1} L(c_1^*) = 0, \quad \ldots, \quad \frac{\partial}{\partial c_k} L(c_k^*) = 0, \quad \frac{\partial}{\partial \lambda} L(\lambda^*) = 0 \qquad (2.6)$$

The conditions above transform to the equations

$$\frac{\partial}{\partial c_1} \delta_f(c_1^*) + \lambda \frac{\partial}{\partial c_1} I(c_1^*) = 0, \ \ldots, \ \frac{\partial}{\partial c_k} \delta_f(c_k^*) + \lambda \frac{\partial}{\partial c_k} I(c_k^*) = 0 \qquad (2.7a)$$

$$I(c_1^*,...,c_k^*) = 0 \qquad (2.7b)$$

Let us assume that the function $I(c_1,...,c_k)$ is invertible for each system parameter. Then we know that

$$\frac{\partial}{\partial c_i} I(c_i) = \frac{\partial}{\partial c_i} I_{sys}(c_i) = \left[ \frac{\partial c_i}{\partial I_{sys}(c_i)} \right]^{-1} \qquad (2.8)$$

and the conditions (2.11) become

$$\frac{\partial}{\partial c_1} \delta_f(c_1^*) \frac{\partial c_1}{\partial I_{sys}} = -\lambda = \ldots = \frac{\partial}{\partial c_k} \delta_f(c_k^*) \frac{\partial c_k}{\partial I_{sys}} \qquad (2.9a)$$

$$I_{sys}(c_1^*,...,c_k^*) = I_0 \qquad (2.9b)$$

The equation (2.9a) says that for the conditions of an optimal information distribution all the terms in (2.9a) should be equal: This is the *principle of optimal information distribution* as it is stated above in section 2.1 and expressed in equation (2.2). The last condition (2.9b) is just our well-known restriction (2.4).

It is well known that the mechanism of Lagrangian multipliers does not provide a general solution, what kind of extremum we have; the decision whether c* is a relative maximum, minimum or a saddle point must be decided according to the application problem. For our case, the decision is clear: According to section (2.1) there exists at least one local minimum. Since we have only one extremum in every application example of section 3, these extrema must be minima.

# 3 Application examples

In this section first we want to demonstrate the procedure above by a very simple example: the approximation of a quadratic form by a polyline or linear splines. Throughout in this example, all design decisions (choice of value ranges etc.) are taken for demonstration purposes only; the whole example is simple enough to be verified analytically by the interested reader.

The section afterwards is intended to be more realistic, but is also more complicated: Here we show the use of the information distribution principle for the application example of a robot control algorithm which uses a non-linear, learned mapping. Since the computations are quite complex, they are given only as an overview. The more interested reader is referred to Brause (1989).
Let us now regard the simplified example.

## 3.1 The approximation of a simple non-linear function

Let us consider the simple non-linear function $f(x) = ax^2 + b$. The approximation of this function can be accomplished by a network with one input x shown in figure 3.1.

**Fig. 3.1** The network for approximating $f(x) = ax^2 + b$ and the unit output function

Another version of the quadratic function is the logistic function $x(t+1)= f(x(t)) := ax(t)(1-x(t)) = ax(t)-ax(t)^2$ which yields deterministic chaotic behavour in the interval [0,1] for some values of the parameter *a*, see e.g. Baker and Gollub, 1990. This system can be approximated by the network of figure 3.1, using an additional, direct input $Wx$ for the second layer to model the linear term $ax$ of the logistic function. The learning of the weights and thresholds by the Backpropagation-Algorithm was demonstrated by Lapedes and Farber, 1987.

Let us return to our example of the quadratic function $f(x) = ax^2 + b$. Each neuron of the network of figure 3.1 has the *output function* $y_i = S(z_i)$ with the *activation function* (potential) $z_i$

$$z_i = \sum_j w_{ij} x_j \qquad (3.1)$$

which becomes for the first layer

$$z_i = w_i x + t_i \qquad \text{with the threshold } t_i \qquad (3.2)$$

and for the second layer

$$\hat{f}(x) = \hat{f}(x,\mathbf{W},T,\mathbf{w},\mathbf{t}) = \sum_i W_i S(z_i) + T \qquad (3.3)$$

with the weight tuples $\mathbf{w}=(w_1,..,w_m)$ and thresholds $\mathbf{t}=(t_1,..,t_m)$ for the first layer and $\mathbf{W}=(W_1,..,W_m)$ and threshold T for the second layer. Let us assume that we use a simple limited linear output function as squashing function

$$S(z_i) = \begin{cases} 1 & 1 < z_i \\ z_i & 0 \le z_i \le 1 \\ 0 & z_i < 0 \end{cases} \qquad (3.4)$$

The definition (3.4) satisfy the conditions $S(\infty)=1$, $S(-\infty)=0$ of Hornik et. al., 1989, and is shown in figure 3.1 on the right hand side. The choice of a linear output function is not only motivated by its analytical simplicity, but also by fact that it can be easily implemented by an ordinary linear electronic amplifier with output signal limits.

Let us assume that all the weights have converged by a proper algorithm for an

approximation of the non-linear function by linear splines. If the linear interval $0<z_i<1$ of each neuron is identical to the others, the superposition will yield again only a line, resulting in a bad approximation of a parabola by one line. To obtain as many approximating lines as possible, the algorithm has to make all intervals different. Since the output of each neuron is only linear in x when $z_i \in [0,1]$ and otherwise it is constant 0 or 1, it is a good choice for the approximation to devide the whole input interval $[X_0,X_1]$ by the *m* neurons of the first layer into *m* equal (see appendix A1) intervals $\Delta x = [x_i-\Delta x/2,\ x_i+\Delta x/2]$ with $x_i=X_0+i\Delta x-\Delta x/2$. The segmented normalized variable $z_i \in [0,1]$ is 1/2 for $x_i$.

In the second layer the output $z_i$ becomes weighted by the weight $W_i$. Together with an offset of the previous intervals it represents the linear part of the approximation function $\hat{f}(x)$ in the interval $[x_i-\Delta x/2,\ x_i+\Delta x/2]$:

$$\hat{f}(x) = \sum_{i=1}^{m} W_i\ S(z_i) + T = \underbrace{\sum_{i=1}^{k-1} W_i + T}_{\textit{offset}} + \underbrace{W_k\ S(z_k)}_{\textit{linear part}} \qquad (3.5)$$

The resulting approximation is shown in figure 3.2. The corresponding values for $w_i$, $t_i$, $W_i$ and T can be easily calculated, see Brause, 1991.

From the conditions of (3.4) we can conclude that the value of $z_i$ at $x_i-\Delta x/2$ is zero and at $x_i+\Delta x/2$ it is one.

Therefore, by (3.2) we get

$$w_i = 1/\Delta x = m / (X_1-X_0) \qquad (3.6a)$$

and

$$t_i = - w_i\ (x_i-\Delta x/2) = X_0/\Delta x +1-i = - mx_i / (X_1-X_0) + 1/2 \qquad (3.6b)$$

Let us choose $W_i$ such that in each segment the spline is parallel to the tangent of f(x) in $x_i$

$$\frac{\partial f(x)}{\partial x}\Big|_{x_i} = \frac{\partial (ax^2 + b)}{\partial x}\Big|_{x_i} = 2ax_i = \Delta y/\Delta x$$

Since the output S(z) is normalized between 0 and 1, we have to choose the weights

**Fig. 3.2** The non-linear function and its approximation

Therefore, the weights become

$$W_i := \Delta y/1 = 2ax_i \Delta x \qquad (3.6c)$$

Then the basic threshold T becomes the offset of the approximation at $X_0$, see figure 3.2. Using (A.1) we get

$$T = f(X_0) - \delta_{lin} = aX_0^2 + b - a/2\,(\Delta x/2)^2 \qquad (3.6d)$$

**Example:**

For a net of *m*:=5 neurons we get for a=1, b=0, $X_0$=-1, $X_1$=1 with $\Delta x$ = 0.4 five non-overlapping intervals [-1,-.6], [-.6,-.2], [-.2,+.2], [+.2,+.6], [+.6,+1] with

$x_i$= {-.8,-.4, 0, +.4, +.8}, $W_i$={-.64, -.32, 0, +.32, +.64},

and $w_i$ = 2.5, $t_i$= {+2.5, +1.5, +0.5, -0.5, -1.5}, T= 0.98.

The maximal linear approximation error $\delta^{lin}$=0.02 has the same order as in the simulation results of Lapedes et al. ,1987.

**Fig. 3.3** The individual neural approximations for a=1, b=0, m=5

In figure 3.3 the superposition of the approximating function by the individual neural output $S_i(x)$ is shown. Each neuron has its linear output restricted to its input interval, otherwise it remains constant.

Due to figure 3.2 (and figure A.1) we might suppose that the error of the approximation does not remain constant, but has minimal and maximal values. This is confirmed in figure 3.4 for the example of 5 neurons.

**Fig. 3.4** The linear approximation error in the interval $x \in [-1,+1]$ for m=5 neurons

In some control applications we are not interested in the mean error over the interval (which is approximately zero in the example above), but in the maximal error that can occur. Thus, we do not aim to minimize neither the average error nor the mean squared error of the approximation, but to minimize the *maximal* absolute error of Eq.(1.1), i.e. the maximal squared error. As the error of the linear approximation we consider therefore the maximal linear approximation error $\delta^{lin}$ which is evaluated in appendix A to

$$\delta^{lin} = a/2(\Delta x/2)^2 \qquad (A.1)$$

This reflects the error due to the finite number of neurons. Let us now consider the other source of the approximation error, the finite information in the weights and thresholds, i.e. the error due to the finite resolutions of the system variables.

### 3.2 The resolution error

To calculate the resolution error due to the number of bits with (1.3) for $w_i$, $t_i$, $W_i$ and T we first have to define the range $V_w$,$V_t$,$V_W$ and $V_T$ of the variables, see Brause (1991). The maximal resolution error $\delta_s$ of a variable *s* in one state is just the half of the resolution increment $d_s$ in equation (1.3)

$$\delta_s = d_s/2 = V_s/2 \; 2^{-I_s} \qquad (3.8)$$

where $I_s$ denotes the number of bits (the information) associated with the variable *s*. In the present approximation function example our information distribution system parameters $c_1$, ...,$c_k$ are represented by the number of bits per variable $I_w$,$I_t$,$I_W$ and $I_T$

and the number *m* of neurons in the first layer. In appendix B the error $\delta^{res}$ due to the finite resolutions $I_w, I_t, I_W, I_T$ and *m* is evaluated to

$$\delta^{res} = 2aX_1 \Delta x [\delta_w X_1 + \delta_t] + m\delta_W + \delta_T \qquad (B.2)$$

### 3.3 The optimal information distribution

As we have already mentioned, we are not interested in minimizing the mean squared error. Besides, since we do not assume anything about the input probability distribution p(x), we can not compute the mean squared error.

Instead, as a performance measure of the approximation network let us compute the maximal absolute error which can occur. The maximal approximation error $\delta_f$ is given by the worst case condition that the maximal linear approximation error $\delta^{lin}$ and the maximal resolution error $\delta^{res}$ do not compensate each other but adding up to

$$\delta_f = \delta^{lin} + \delta^{res} \qquad (3.9)$$

The whole information $I_{sys}$ contained in the network is the sum of the information $m(I_w+I_t)$ of the *m* weights and thresholds in the first layer and the information $mI_W+I_T$ of the *m* weights and the threshold in the second layer

$$I_{sys} = m(I_w+I_t+I_W) + I_T \qquad (3.10)$$

When we add some information to the system by augmenting the number *m* of neurons, the resulting approximation will be better and, naturally, the approximation error will diminish. When we add some neurons, but reduce the information in the weights and threshold, such as to conserve the overall system information, the result is

not so clear. In figure 3.5 the approximation error is shown on a logarithmic scale for different values of *m* and constant system information $I_{sys}$=708.45 bits; the number of bits for all other variables are the same $I_w=I_t=I_W= I_T$ and can be directly computed by equation (3.10).

**Fig. 3.5** The approximation error at constant system information (a=1, b=0)

The minimal error of $\delta_f=2.28x10^{-3}$ is at m*=16.2 neurons and $I_T$=14.2 bits, about 3% worse than with the optimal system parameters (see example ahead). To get the optimal parameters, we just have to compute the conditions for the multivariate minimum of $\delta_f(m,I_w,I_t,I_W,I_T)$ which we have already solved in section 2.1 and 2.2.
The condition (2.2) for an optimal information distribution becomes

$$\frac{\partial}{\partial m}(\delta^{lin} + \delta^{res}) \left(\frac{\partial I_{sys}}{\partial m}\right)^{-1} = \ldots = \frac{\partial}{\partial I_T}(\delta^{lin} + \delta^{res}) \left(\frac{\partial I_{sys}}{\partial I_T}\right)^{-1} \qquad (3.11)$$

with the derivatives of (3.10)

$$\frac{\partial I_{sys}}{\partial m} = I_w+I_t+I_W \qquad \frac{\partial I_{sys}}{\partial I_w} = m = \frac{\partial I_{sys}}{\partial I_t} = \frac{\partial I_{sys}}{\partial I_W} \qquad \frac{\partial I_{sys}}{\partial I_T} = 1 \qquad (3.12)$$

The 5 terms of (3.11) should be are all equal, giving us 4 equations with 5 variables. In Brause, 1991 this is evaluated giving us the three equations

$$I_t = I_w + C \qquad \text{with } C:= \log_2((X_1-X_0)/X_1) \qquad (3.13)$$

$$I_W = I_t + C \qquad (3.14)$$

$$I_T = I_W + \log_2(g_T(m)/2) - \log_2((X_1-X_0)^2/m) \qquad (3.15)$$

and the equation for the number of neurons

$$m = h(m,I_T)^{1/3} \qquad (3.16)$$

This we can use for numerically given $I_T$ as an iteration formula at the (s+1)-th iteration for *m*:

$$m(s+1) = h(m(s),I_T)^{1/3} \qquad (3.17)$$

Since the derivative of $h(m)^{1/3}$ is lower 1, the convergence condition is satisfied and the iteration converges.

*Example*

Let us choose an information of 16 bit in the threshold T. Therefore, in the simple case of $X_0$=-1, $X_1$=+1, a=1, b=0 we have with $I_T$ = 16 bit the optimal configuration at m=16.54 neurons, $I_W$ =14.95 bit, $I_t$=$I_W$-C =13.95 bit and $I_w$=$I_t$-C=12.95 bit. The overall information in the network with Eq. (3.10) is then $I_{sys}$ = 708.45 bits (the same $I_{sys}$ as in the example above) and the overall approximation error is $\delta_f = 2.213\times10^{-3}$ with the pure linear approximation error part of $\delta^{lin}=1.83\times10^{-3}$. If we augment the information capacity of the system and use $I_T$=32 Bit, the overall error will diminish to $\delta_f =1.847\times10^{-6}$ when we use the optimal system parameters.

The example of the approximation of a simple quadratic function is quite instructive to evaluate, but has the disadvantage that it is not very common in real world applications. The question is, whether the proposed principle of information distribution works in a more realistic environment.

### 3.4 The approximation of robot manipulator control

For this purpose let us consider the more complicated task of robot manipulator position control. The *kinematic* control computes the Cartesian position **x** of the end point of a robot manipulator, composed of several segments and joints, by a straightforward matrix multiplication (*homogeneous transformation*) between all segment-matrices when the joint coordinates (joint angles) $\theta_i$ are given. The inverse transformation, the *inverse kinematics*, does the inverse task: when the absolute

Cartesian coordinates **x** of the end point (e.g. the palm of the robot hand) is given, it computes the appropriate joint coordinate $\theta_i$ for each segment.

The inverse kinematic is a quite complicated function and not easy to find. When the rotational axes of the joints are oriented not in parallel or orthogonal, it is very hard or quite impossible to find an analytical solution. This fact prohibits the exploration of user-defined robot architectures and limits the adaption of robot architectures to the user's needs.

A very promising approach is to *learn* the non-linear mapping of inverse kinematic. One of the existing approaches by neural network systems is the use of Kohonen's, 1984 Topology-conserving maps by Ritter, Martinetz and Schulten, 1989. Since the mapping is very coarse for a small amount of neurons, they additionally use a linear approximation with learned coefficients. In figure 3.6 a neural network for the robot control is shown which models the main steps of the algorithm.

**Fig. 3.6** An approximator network for robot control

The whole workspace {**x**} of the robot is devided into *m* segments, each one defined by a $\mathbf{w}^k$, the center of the workspace segment *k*. Instead of one exclusively activated neuron *c* in the Kohonen net which has the smallest distance $|\mathbf{x}-\mathbf{w}^c|$ there is a cluster of three neurons which becomes active when their segment of the workspace is concerned. This cluster have to generate the 3-dim. difference vector $\mathbf{x}-\mathbf{w}^c$ for the linear approximation in the workspace segment by the second layer. By the learning rules, the constant input unit of each cluster (black dot in fig.3.6) generates a constant term $\Theta^c=(\theta_1^c,\theta_2^c,\theta_3^c)$ which is the vector of the angles corresponding to $\mathbf{w}^c$.

Thus, we have a two-layer approximation network again. Since the performance of this approach heavily depends on the resolution of the neural net and the resolution of the internal representation, we have to apply our methods of section 2 to prevent an exhaustive need for storage. Instead of a one-dimensional problem with *m* neurons the number of neurons grow by $m=n^3$ having *n* neurons per dimension. Here we have to balance the number *n* of storage cells (number of neurons) against the bits per cell (resolutions $I_w$, $I_\theta$, $I_A$ of the weights and coefficients). The choice for the system parameters *n*, $I_w$, $I_\theta$, $I_A$ can be done by the information distribution principle introduced above most efficiently.

For this purpose, let us assume that the stochastic approximation process of the Kohonen mapping has become stable and the mapping has perfectly converged. Nevertheless, there remains an error due to the discrete approximation of the non-linear function. For the example of the commonly used PUMA robot (figure 3.7) this was evaluated in Brause, 1989, based on the strategy for optimal storage

**Fig. 3.7** The PUMA robot (after Fu, Gonzales and Lee, 1987)

distribution. The main results are given below.

Let us first evaluate the maximal error $\delta^{lin}$ due to the linear approximation. Since we have only rotational axes in the system, one of the most difficult and important tasks for the manipulator is a linear, straight movement as it is often required in applications. Therefore, we consider the error on a straight line through the whole cubic work space of the manipulator. This resembles a cut through the error-weighted workspace. The numerically computed approximation error is shown in figure 3.8(a) on the left hand side. The parameter of the approximation error is *n*, the number of neurons in one dimension. Since the robot works in three dimensions,

we have $m=n^3$ neurons in the whole system.

Interestingly, the lines of the different parameter values n=10, 100, 1000 seem to be shifted vertically with the same offset. A numerical evaluation of the error on the positioning point with the maximal error (approximately at the third path point) shows us that this is right; in figure 3.8(b) on the right hand side the logarithm of the joint error is drawn versus the number *n* of the neurons in logarithmic scale. This gives us the analytical expression of $\delta^{lin}=cn^b$ as a good approximation with numerical obtained values for *c* and *b*. This coincidences well with the analytical expression (A.2) for the linear approximation error of the example of a quadratic function.

(a) (b)

**Fig. 3.8** (a) The absolute positioning error and (b) the joint error as a function of the number of neurons per dimension

The resolution error $\delta^{res}$ of the linear approximation scheme can be easily calculated by the same ideas as for equation (B.2).

The maximal error is, again, the superposition of the error of the linear approximation and the resolution error

$$\delta_f = |\underline{\delta}^{lin} + \underline{\delta}^{res}|$$

with $\underline{\delta}^{lin}$ and $\underline{\delta}^{res}$ denoting the error vectors. Since the form of both errors are now analytically known, the conditions for the optimal information distribution of equation (2.2) can be calculated, using the derivatives of $\delta_f$, i.e. of $\delta^{lin}$ and of $\delta^{res}$. Of the resulting three conditions for four parameters all can analytically be solved except the condition for *m*, which was numerically iteratively approximated. The optimal system parameter values for a fixed amount of system information are shown in figure 3.9(a) on the left hand side.

(a) (b)

**Fig. 3.9** (a)The optimal parameter configurations and (b) the corresponding Cartesian error for minimal information storage

Now we have an optimal configuration of all system parameters yielding the minimal possible information storage amount for a given Cartesian error. The Cartesian error as a function of this optimal storage is shown in figure 3.9(b) on the right hand side for the situation when all weights and thresholds are forced to have the same resolution (number of bits per variable) but optimal *n* and, furthermore, when they all have different, optimal resolutions.

For a reasonable error of 0.2 mm, a value which is in the range of normal mechanical inaccuracy of the PUMA manipulator, the necessary 1.9 MB of storage memory is contained in $m=39.6^3$ neurons and a constant resolution of $I_w=16.4$ Bits for all weights. The optimal configuration with different weight resolutions gives only a 18% smaller error, and therefore do not encourage the use of multiplication operations with variable accuracy which would be necessary in this case.

It should be noted that figure 3.9 assumes real-valued number of bits and neurons. Certainly, in real applications we must use integer values (truncated or rounded) for all parameters which will result in a slightly different optimum and increased approximation error. The best selection will choose of the possible integer tupels $(I_w,m^*)$ the one with the smallest error $\delta_f$.

Experiments with a computer-simulated neural network controlling a real PUMA-like robot confirm the considerations above .

# 4 Conclusion

The error-bounded descriptional complexity of approximator networks is determined by the principle of optimal information distribution. This is a criterium for the efficient use of the different information storage resources in a given network.

Furthermore, it can be used as a tool to balance the system parameters and to obtain the optimal network parameter configuration according to the minimal usable storage amount for a maximal error which is given.

In this paper two examples are presented. First, a simple non-linear function approximation is evaluated, the conditions for optimal system configuration are stated, their solutions are analytically computed and their nature is explained. Second, the more complicated function of the inverse kinematic of a PUMA robot is considered and the results for optimal system parameters, which are partially obtained by numerical iterative approximations, are shown.

The benefits of the proposed method are not limited to real networks, but apply also to all computer simulations. Here we have a tool to tailor the storage requirements according to the application needs in an optimal way.

## References

Baker, Gregory and Gollub, Jerry (1990). **Chaotic dynamics: an introduction** Cambridge University Press.

Brause, R. (1989). Performance and Storage Requirements of Topology-Conserving Maps for Robot Manipulator Control. *Internal Report 5/89*, Fachbereich Informatik, University of Frankfurt, FRG.

Brause, R. (1991). Approximator Networks and the Principle of Optimal Information

Distribution. *Internal Report 1/91*, Fachbereich Informatik, University of Frankfurt, FRG,
and *Proc. ICANN-91* Helsinki, Elsevier Publ. Comp., North-Holland.

Gallager, R. (1968). **Information Theory and Reliable Communication**.
New York, London, Sydney, Toronto: John Wiley and sons, inc.

Girosi, F. Poggio,T. (1990). Networks and the Best Approximation Property.
*Biolog. Cybern.*, **63**,169-176.

Hornik, K., Stinchcomb, M., White,H. (1989). Multilayer Feedforward Networks are Universal Approximators, *Neural Networks*, **2**, 359-366.
and
Stinchcomb, M., White,H. (1989). Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. *Proc. Int. Joint Conf. Neural Networks,* Washington DC, pp. I/607-611

Fu, K.S., Gonzales,R.C., and Lee,C.S. (1987). **Robotics**, McGraw Hill Book Comp.

Kohonen,T.(1984). **Self-Organisation and Associative Memory**.
Berlin, New York, Tokyo: Springer Verlag.

Kolmogorov, A.N., Tihomirov,V.M. (1959). ε-Entropy and ε-Capacity of Sets in Functional Spaces, *Uspehi Mat (N.S.)*, **14**, 3-86. In: *Am. Math. Soc. Transl.*(1961), Series 2, **17**, 277-364.

Lapedes,A., Farber,R. (1987). Nonlinear Signal Processing using Neural Networks:

Prediction and System Modelling; *Los Alamos preprint* LA-UR-87-2662

Li,Ming, and Vitanyi,Paul (1990). Kolmogorov Complexity and its Applications. In: J. van Leeuwen (Ed.) **Handbook of Theoretical Computer Science**, Elsevier Sc. Publ.

Ritter,H., Martinetz,T., Schulten,K. (1989). Topology-Conserving Maps for Learning Visuomotor-Coordination, *Neural Networks*, 2/**3**, 159-167. New York: Pergamon Press.

Shannon, C.E., Weaver ,W. (1949). **The Mathematical Theory of Information**, Urbana: University of Illinois Press.

Williams, Robert (1991). ε-Entropy and the Complexity of Feedforward Neural Networks. In R.Lippmann, J.Moody, D. Touretzky (Eds.): **Advances in Neural Information Processing Systems 3**. San Mateo, CA: Morgan Kaufmann Publ.

# Appendix A: The linear approximation error

The non-linear function in the intervall $[x-\Delta x/2, x+\Delta x/2]$ is

$$f(x) = ax^2 + b$$

and the linear approximation by the neural network is

$$\hat{f}(x) = \alpha x + \beta \qquad \text{with } \alpha := 2ax$$

**Fig. A.1** The error of the linear approximation

The approximation error $\delta_l$ is (see figures 3 and A.1 above)

$$\delta_l(x) = f(x) - \hat{f}(x) = ax^2 + b - 2axx - \beta = b - \beta - ax^2 =: d$$

$$\delta_l(x_i+\Delta x/2) = f(x_i+\Delta x/2) - \hat{f}(x_i+\Delta x/2) = d + a(\Delta x/2)^2$$
$$\delta_l(x_i-\Delta x/2) = f(x_i-\Delta x/2) - \hat{f}(x_i-\Delta x/2) = d + a(\Delta x/2)^2$$

The errors at the borders are equal. The maximal error $\max(|\delta_l(x_i)|,|\delta_l(x_i+\Delta x/2)|)$ is minimal when the errors are equal

$$|\delta_l(x_i)| = |\delta_l(x_i+\Delta x/2)| \qquad \text{or} \qquad |d| = |d + a(\Delta x/2)^2| \qquad (A.0)$$

This is given when

$$d = -a/2(\Delta x/2)^2$$

Therefore, the *maximal linear error* $\delta^{lin}$ depends not on the value of $x_i$, it is the same in the whole interval

$$\delta^{lin} = a/2(\Delta x/2)^2 \qquad (A.1)$$

Since we have $\Delta x = (X_1-X_0)/m$ we get

$$\delta^{lin} = m^{-2}a(X_1-X_0)^2/8 = c\, m^b \qquad (A.2)$$

with $c = a(X_1-X_0)^2/8)$ and $b = -2$

**Note**: For this approximation problem, the maximal approximation error is minimized when we divide the whole interval $[X_0,X_1]$ into *m* equal segments. This can easily be proven by the following:

Let us regard the interval segment which has the maximal approximation error. According to equations (A.0), the maximal linear error depends not on the value of $x_i$, the middle point of the i-th interval segment, but only on the length $\Delta x_i$ of the segment. Thus, all the segments can be sorted into a descending order of both their length and their associated error.

Now, if we reduce the segment length $\Delta x_i$ and increase the length $\Delta x_k$ of the next segment in the order, the maximal error diminishs until it becomes equal to the error of the next segment. Then both segment lengths and errors are equal. A further reduction of $\Delta x_i$ alone will not change the maximal error, we have to reduce both the segment lengths $\Delta x_i$ and $\Delta x_k$, and have to increase the length of the third segment in the order until all three errors and segment lengths become equal.

Let us assume that this is true for the *n* first segments in the initial order. Then the idea above is also valid for the n+1-th reduction step to the n+1-th segment: by complete induction all segments have to be equal for the minimum of the maximal error, given in equation (A.1).

## Appendix B: The resolution error

For the computation of the resolution error let us assume that in all weights and thresholds the maximal increment error $\delta$ has occurred. The resolution and therefore the maximal increment error in one variable should be independent of its index. Then the approximating function becomes with (3.2) and (3.3)

$$\hat{f}(x,\delta) = \Sigma_i\ (W_i + \delta_W)\ S(z_i+\delta_z) \quad + T+\delta_T \qquad \text{(B.1)}$$
$$= \Sigma_i\ W_i\ S(z_i+\delta_z)\ + T + \Sigma_i\ \delta_W\ S(z_i+\delta_z)\ +\delta_T$$

Because the intervalls are exclusive, for the k-th intervall we have to regard only the influence of one neuron of the first layer; for i<k we have $S(z_i) = S(z_i+\delta_z) = 1$ and for i>k we have $S(z_i+\delta_z)=0$.

$$\hat{f}(x,\delta) = (\Sigma_i^{k-1} W_i)\ + W_k\ S(z_k+\delta_z) + T + (\Sigma_i^{k-1}\delta_W) + \delta_W\ S(z_k+\delta_z) +\delta_T$$
$$= \hat{f}(x)\ + W_k\ \delta_z + (k-1)\delta_W + \delta_W\ S(z_k+\delta_z) +\delta_T$$

The *maximal error* $\delta^{res}$ is encountered at $\max(x) = X_1$ on the boarder of the intervall $[X_0,X_1]$. The contribution of the term $\delta_W S(.)$ becomes maximal $\delta_W$ when $S(.) = 1$. Therefore, we have

$$\hat{f}(X_1,\delta) = \hat{f}(X_1) + (m-1)\delta_W + W_m\delta z_m + \delta_W\ S(z_m+\delta_z) +\delta_T$$
$$= \hat{f}(X_1) + m\delta_W + W_m\delta_z\ +\delta_T$$

and so with $\delta_z=\delta_w x_m+\delta_t$ we get

$$\delta^{res} = \hat{f}(X_1,\delta) - \hat{f}(X_1)= m\delta_W + W_m\ (\delta_w\ X_1+\delta_t) + \delta_T$$

With (3.6c) we get

$$\delta^{res} = 2aX_1\ \Delta x\ [\delta_w\ X_1+\delta_t] + m\delta_W + \delta_T \qquad \text{(B.2)}$$

# Figures and Legends

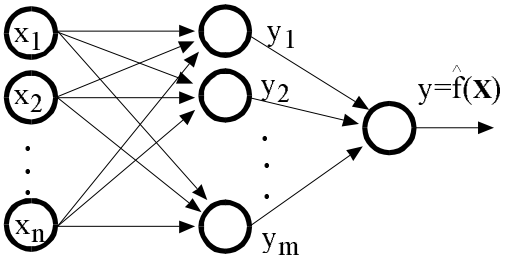


**Fig. 1** A two-layer universal approximation network.

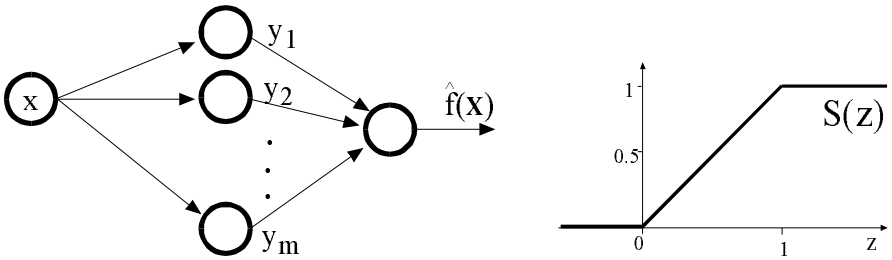


**Fig. 3.1** (a) The network for approximating the function $f(x) = ax^2 + b$.
(b) the limited linear neural unit output function S(z). As you can see, it satisfies the conditions $S(-\infty)=0$ and $S(\infty)=1$.

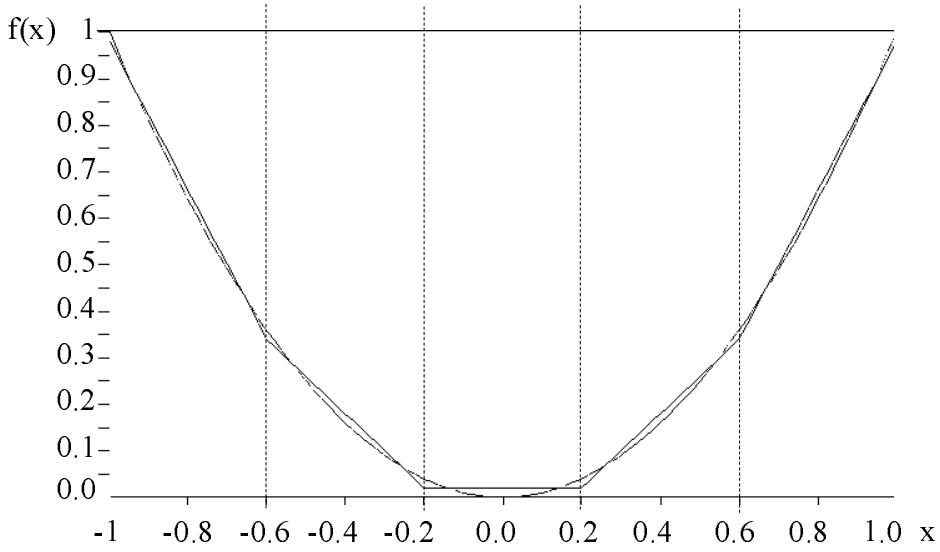


**Fig. 3.2** The non-linear function f(x) and its approximation by linear splines, the polygon of the superposition of the linear neuronal output. The dotted vertical lines denote the input interval borders for each neuron.

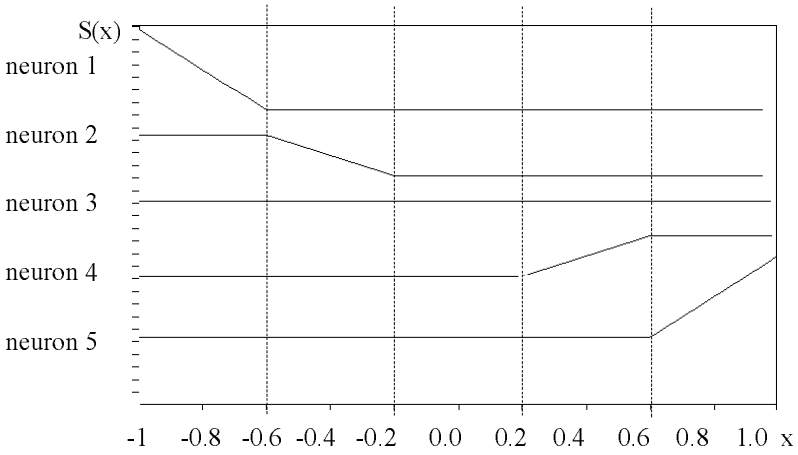


**Fig. 3.3** The individual neural approximation outputs for a=1, b=0, m=5. By the dotted lines the input is devided in exclusive intervals, again. In each input interval, there is a neuron with a linear output. The weighted superposition of all outputs becomes a polygon. For the sake of clarity the output response of the neurons are shifted vertically in the drawing.

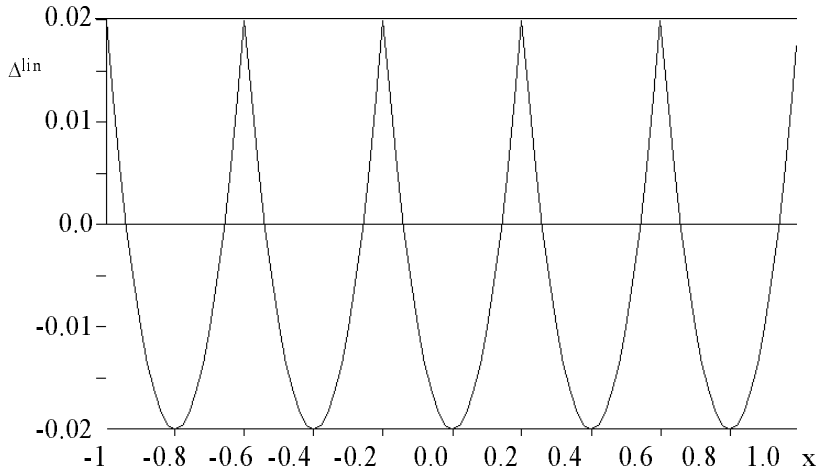


**Fig. 3.4** The approximation error $\delta^{lin}$ in the interval $x \in [-1,+1]$ for m=5 neurons. As you can see, the error is maximal in the middle and at the borders of the intervals.

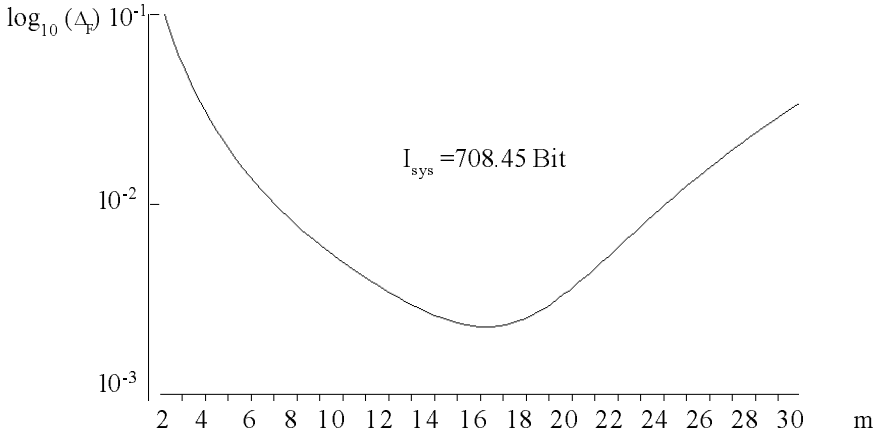


**Fig. 3.5** The maximal approximation error $\delta_f$ at constant system information $I_{sys}$. The error is drawn on a logarithmic scale.

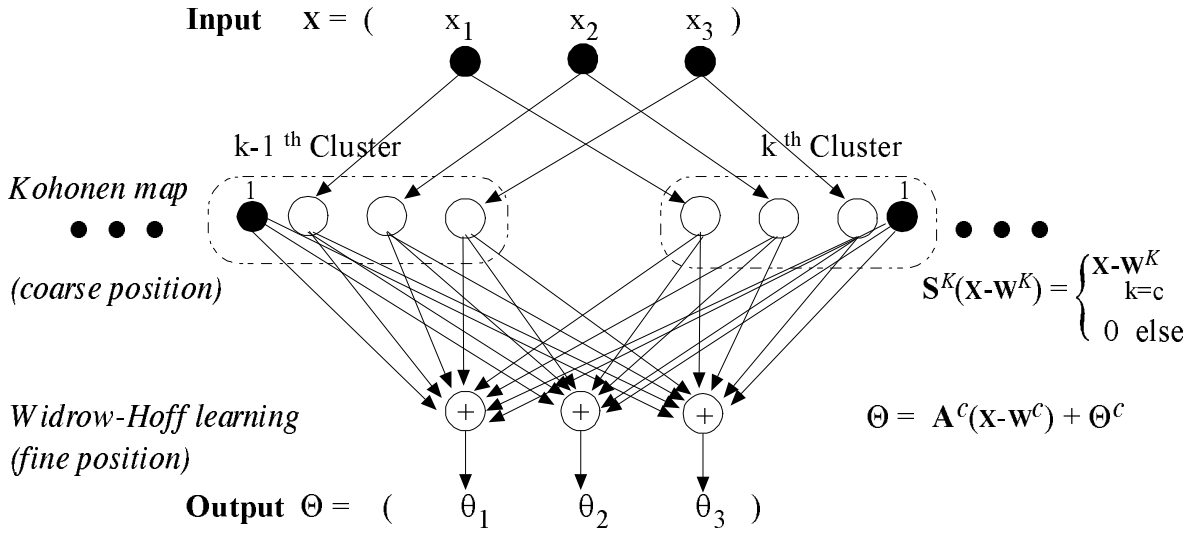


**Fig. 3.6** A two-layer approximator network for the inverse kinematic control of robots. Each cluster responds exclusively only in one workspace segment (winner-take-all network) with one set of learned joint angles corresponding to the center of the workspace segment. Additionally, the second layer interpolates linear this non-linear mapping by the learned matrix $A^k$.

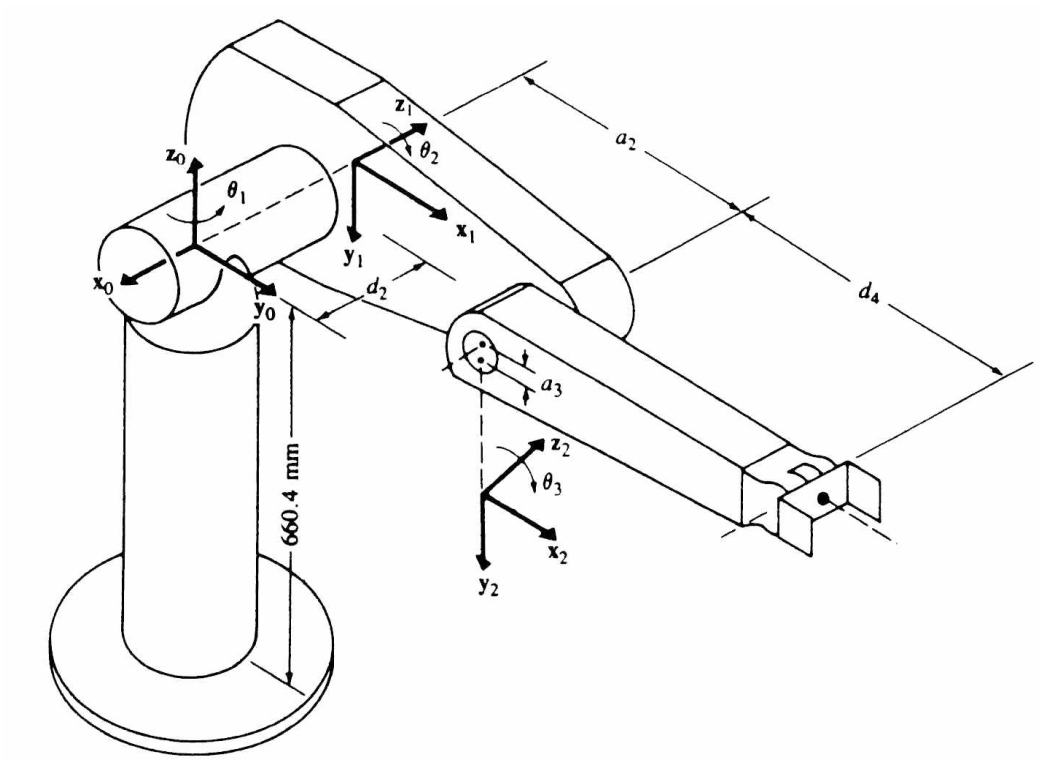


**Fig. 3.7** The PUMA robot (after Fu, Gonzales & Lee, 1987). The palm of the robot hand which is subject to the inverse kinematic is marked by a dot.

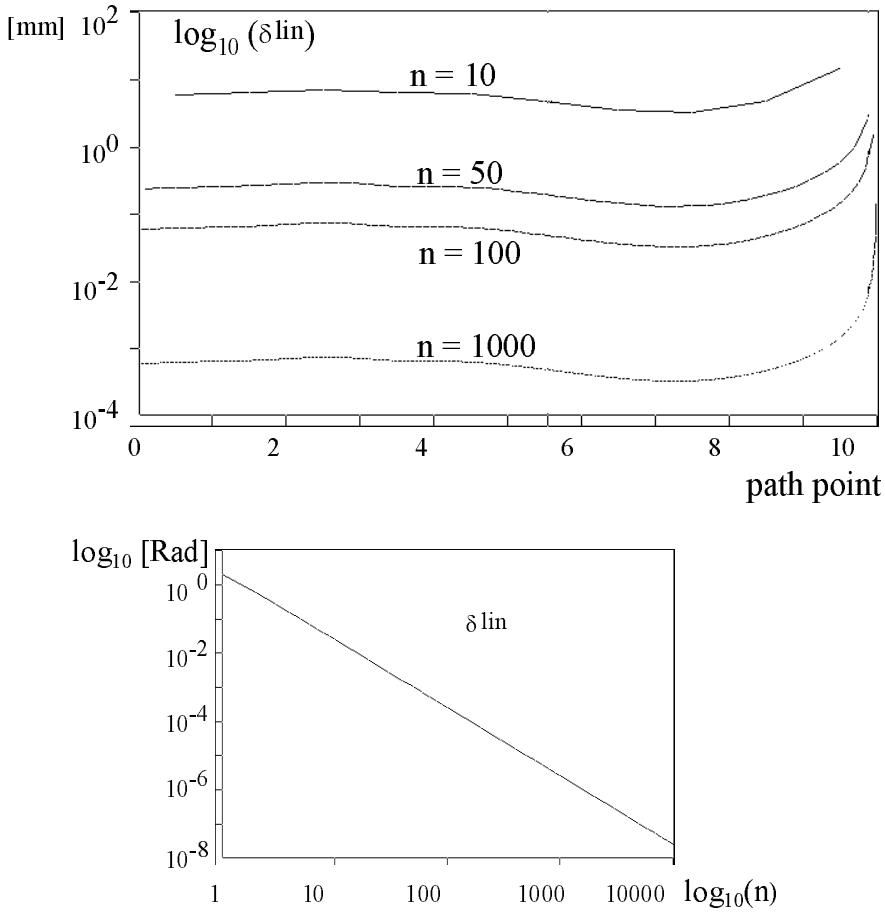


**Fig. 3.8** (a) The absolute Cartesian positioning error due to the linear approximation on a linear path through work space, shown on a logarithmic scale,
(b) the joint error due to the linear approximation as a function of the number of neurons per dimension. Both axes are logarithmically scaled.

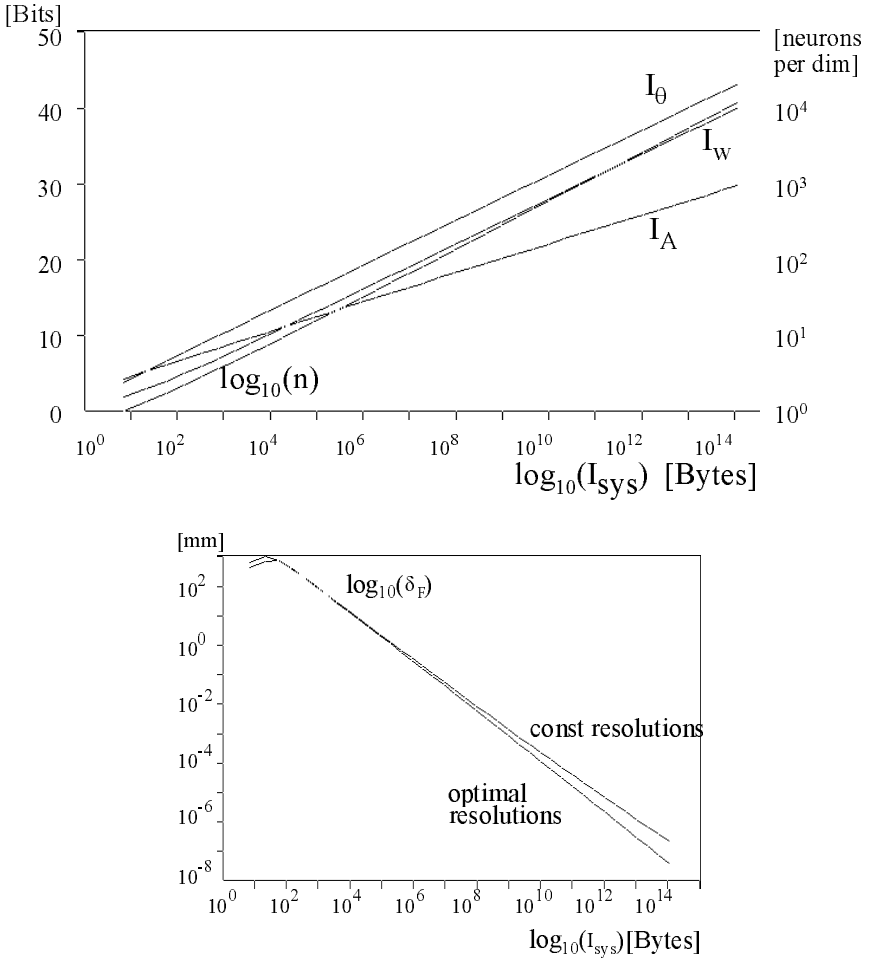


**Fig. 3.9** (a) The optimal values for the network parameters when the available system information is given.
and (b) the resulting Cartesian error at an optimal information distribution. For the same amount of information, the approximation error of two possible configurations are drawn in a logarithmic scale: One with all weights having the same resolution and one with different, optimally computed resolutions. The optimal number $m$ of neurons results as a function of the other parameters. For huge storage sizes, the difference between the two approaches is remarkable, but for moderate, more realistic requirements the difference can be neglected.

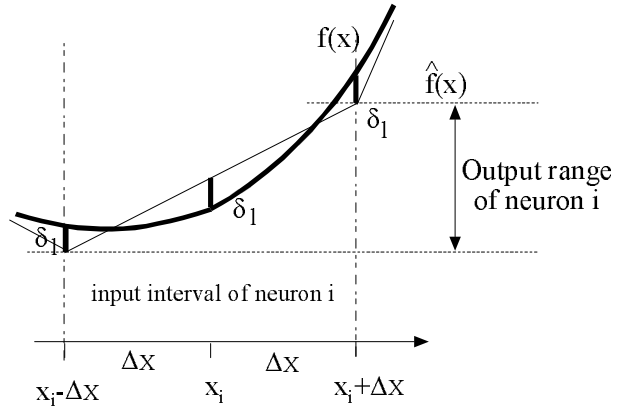


**Fig. A.1** The error of the linear approximation of a quadratic function. Remark that the errors occur in the middle and at the borders of the interval.