DETERMINATION OF NEURAL NETWORK PARAMETERS
BY INFORMATION THEORY

RÜDIGER W. BRAUSE,
University of Frankfurt, FB Informatik,
Postbox 11 93 2, D-6000 Frankfurt 11, FRG.

Received 11 February 1992

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 rests to determine the necessary number of neurons and the necessary accuracy (number of bits) per weight for a satisfactory operation.

In this paper the accuracy of the weights and the number of neurons are seen as general system parameters which determine the maximal output information (i.e. the approximation error) by the absolute amount (network description complexity) and the relative distribution of information contained in the network. A new principle of optimal information distribution is proposed and the conditions for the optimal system parameters are derived.

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 system parameters, i.e. the number of neurons and the different resolutions of the variables.

Keywords: Transinformation, information distribution, approximation network, robot control error, storage optimization.

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 discrete electronic components.

Similar to the well-known theorem of Stone-Weierstraß (see e.g. [4] for regularization networks) Hornik, Stinchcomb and White have shown [14], [6] that every function can be approximated by a two layer neural network (see Fig. 1) when a sufficient large number \( m \) of units is provided. Sufficient 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 one-dimensional output, as it was done in the paper [6]. Analogous results hold for multi-output networks, i.e. vector-valued functions.
1.1 The representation of information in neural networks

To give an answer to this question, we first have to remark that our standard modelling of artificial neural nets do not reflect an important feature of reality: the discreteness of all real valued events. Contrary to the modelling of synaptic weights and neuronal activity (spike-frequency) by real numbers, there do not exist real numbers in reality.

Instead, there exist a kind of noise and imprecise operations which give rise to a certain amount of error in all real world systems. Especially in simulations and implementations of neural nets we replace all real numbers by more or less fine-grained physical variables, e.g. counters or other discrete variables, with a finite error. This concept is consistent with the restriction of "finite information" in our system: the information of a variable $x$ is defined by

$$ I(x_i) := - \log (P(x_i)) \quad \text{[Bits]} $$  \hspace{1cm} Information \hspace{1cm} (1.1)

If all states $x_i$ are equiprobable, the information is the binary logarithm of the number of possible states. For a real number, the number of different values $x_i$ is infinite. Thus, if we have no a priori knowledge about the occurrence of the states and we have therefore to assume a uniform, non-vanishing probability distribution for them, a real number has an infinite amount of information. This argument is also valid for the averaged information, the entropy, introduced by Shannon [13]

$$ H := \langle I(x) \rangle = - \sum i P_i \log P_i = - \int p(x) \log p(x) \, dx, \quad Entropy \hspace{1cm} (1.2) $$

which also becomes infinity for a uniform distribution $p(x) := 1/d$ over the whole range of the real variable $x$

$$ \lim_{d \to \infty} H(d) = \lim_{d \to \infty} -d^{r_d/2} \frac{1}{d} \log (1/d) \, dx = \lim_{d \to \infty} - \log (1/d) = \infty. $$

Because all systems deal with finite amounts of information, there are no "real" real numbers used in neural systems; all weights have a distinguishable number of states (at least due to quantum physics) and therefore contain a certain amount of information in the sense of the above definition (1.1).
1.2 Optimal Approximation Layers

Many technical and biological systems consist of stages or layers of operations, which process the incoming information in a pipe-lined manner. If we assume the necessity of all stages, then we can optimize the whole information processing system when we optimize each layer separately. Therefore, let us consider the conditions for optimal layers.

This leads us to the question: optimal - in what sense?

All feed-forward layers can be seen as a mapping of a set of points \( \{x\} \) of the input space to discrete points \( \{y_i\} \) of the output space. If there is only a single point in the output space, the approximation will not be fine: there is certainly less information in the output than in the input. Therefore, one plausible principle of a good mapping is to transmit as much information from the input to the output as possible (\textit{maximal information} principle). This optimality criterion was proposed for instance by Linsker [10] for neural networks, who suggested that this might be a fundamental principle for the organization of biological neural systems, and Haken [5] who found this a common principle in physical and chemical systems. Originally it was introduced by Shannon [13] for the transmission channel of a message between a sender and a receiver. In Fig. 2, this situation is shown for one layer.

![Fig. 2. The information transmission through a layer.](image)

Knowing the input pattern \( x \), the Shannon information gain from the \( N \) output points \( y_i \) is by (1.1)

\[
I_{\text{trans}} := I_{\text{out}} - I_{\text{out/inp}} = -\ln[P(y_i)] + \ln[P(y_i/x)] .
\]

The average transmitted information or \textit{transinformation} \( H_{\text{trans}} \) for all inputs and outputs is

\[
H_{\text{trans}} := \langle I_{\text{trans}} \rangle_{y_i,x} = \langle I_{\text{out}} \rangle_{y_i,x} - \langle I_{\text{out/inp}} \rangle_{y_i,x} = -\sum_i P(y_i) \ln[P(y_i)] - \sum_x P(x) \sum_i P(y_i/x) \ln[P(y_i/x)] .
\]

The transinformation \( H_{\text{trans}} \) is maximized when

\[
\langle I_{\text{out}} \rangle_{y_i,x} \overset{!}{=} \max ,
\]

\[
\langle I_{\text{out/inp}} \rangle_{y_i,x} \overset{!}{=} \min .
\]

The condition (1.3) results when
This is shown for the convenience of the reader in appendix A. If we identify each state $y_i$ with a small, finite interval in the output continuum, the condition above says that we should partition the most frequently used regions of the output space by a finer grid to reflect the non-linear properties of the mapping, contrary to the findings of Linsker [11]. This is an important result for neighborhood-conserving mappings as they are used in section 3.4.

When we do not know the input distribution, we might assume an uniform probability distribution at the output and condition (1.5) already holds. Then the demand of (1.3) transforms to the demand for maximizing the number of distinguishable output states. This is done in the next section 2.

Let us now consider the second condition. For the demand of (1.4) we know now that the values for $P(y_i|x)$ must be very unequal to yield a minimum. This is the case when every input pattern $x$ is assigned deterministically to only one appropriate state $y_i$ and the noise (see Fig. 3) is set to zero. With this assumption, we get $\langle \text{out/in} \rangle = 0$ (see [2]), which is the absolute minimum for the information loss.

Under this condition, it is sufficient for an optimal layer to supply the demand of (1.3) for maximal output information. The next section shows us, how we can obtain this by a proper choice of the network parameters.

2. Optimal Information Distribution

An important example for a feed-forward network layer is the approximator network of Fig. 1. Let us regard an approximation $\hat{f}$ for the function $f: \mathbb{R}^n \ni x \rightarrow f(x) \in \mathbb{R}$. For example, this can be done by a two-layer neural network (Fig. 1). Let the positive root of the maximal quadratic error of this approximation be $d_f$ with

$$d_f^2 = (\hat{f}(x) - f(x))^2.$$  \hspace{1cm} (2.1)

Then we can regard the error as a kind of discretization error. Denoting the complete value range with $V_f := |f_{\text{max}} - f_{\text{min}}|$, we can conclude that there are only $V_f/d$ distinguishable, fixed states of the variable $f$ which differ by an increment of $d = 2d_f$. All other states are indistinguishable from deviations of the fixed states. Thus, unless we do not know anything more about the input distribution of $\{x\}$ and therefore nothing more about the error distribution, the output has minimal

$$I_{\text{out}} = \log_2 \left( \frac{V_f}{d} \right)$$  \hspace{1cm} (2.2)

bits of information.

The system parameters which determine the error of the approximation, are on the one hand the resolution of the weights or its information content

$$I_w = \log_2 \left( \frac{V_w}{d_w} \right)$$  \hspace{1cm} (2.3)

with the weight increment $d_w$ 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 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. Fig. 7.

Therefore, we have to solve the problem: what is the best information distribution in the network, i.e. what is the best choice for the parameters \( m \) and \( I_w \) to maximize the Information \( I_{\text{out}} \) or to minimize the approximation error \( d_e \), using a fixed amount of system information \( I_{\text{sys}} \)? Let us denote the parameters \( m, I_w, \ldots \) as general system parameters \( c_1, \ldots, c_k \).

### 2.1. The Principle of Optimal Information Distribution

Let us first derive the conditions for the optimal system parameters by some plausible considerations, first presented in [2]. The conventional mathematical approach will be covered by the section 2.2 later.

Assume on the one hand that we transfer a fixed, small amount of information from one parameter to another and we will find the maximal output information \( I_{\text{out}} \) increased by decreasing the approximation error. In this case the information distribution induced by the parameter values of \( c_1, \ldots, c_k \) was not optimal; the new one is better. Let us assume that on the other hand we find that the output information has decreased, then the information distribution is not optimal, too; by making the inverse transfer we can also increase \( I_{\text{out}} \).

These considerations lead us to the following extremum principle:

In an optimal information distribution a small (virtual) change in the distribution (a change in \( c_1, \ldots, c_k \)) neither increases nor decreases the maximal output information.

A small increment of additional information \( \delta I_{\text{sys}} \) in the system will produce a change \( \delta I_{\text{out}} \) in the minimal output information

\[
\delta I_{\text{out}} = \delta I_{\text{sys}} \frac{\partial I_{\text{out}}}{\partial I_{\text{sys}}} = \delta I_{\text{sys}} \sum_{i=1}^{k} \frac{\partial}{\partial c_i} I_{\text{out}}(c_1, \ldots, c_k) \frac{\partial c_i}{\partial I_{\text{sys}}}.
\]  

(2.4)

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

With the definition (2.2) we get for each term of the sum of (2.4)

\[
\frac{\partial}{\partial c_i} I_{\text{out}}(c_1, \ldots, c_k) = \frac{\partial}{\partial c_i} (\log(V_i) - \log(d)) = - \frac{1}{d} \frac{\partial d}{\partial c_i} = - \frac{1}{d} \frac{\partial d_e}{\partial c_i}.
\]

(2.5)

and so the optimal distribution resides when

\[
\frac{\partial d_e}{\partial c_1} \frac{\partial c_1}{\partial I_{\text{sys}}} = \ldots = \frac{\partial d_e}{\partial c_k} \frac{\partial c_k}{\partial I_{\text{sys}}}.
\]

(2.6)
is satisfied. The \( k \) independant terms gives us \((k-J)\) equations fo \( k \) variables \( c_1, \ldots, c_k \), leaving us with a degree of freedom of one. So, choosing the amount of available information storage \( I_{sys}(c_1, \ldots, c_k) := I_0 \), the parameters \( c_1, \ldots, c_k \) are fixed and with \( I_{out} \) the smallest error \( d_t \) 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. The Optimal System Parameters

Now we want to compare the above principle to a more conventional mathematical approach. The maximal information \( I_{out} \) introduced above is a multivariate function \( I_{out}(c_1, \ldots, c_k) \). If we want to get the maximal information out of the system using only a certain amount of system information we look for an optimal parameter tuple \((c_1^*, \ldots, c_k^*)\) such that

\[
I_{out}(c_1^*, \ldots, c_k^*) = \max_{c_1, \ldots, c_k} I_{out}(c_1, \ldots, c_k) \tag{2.7}
\]

which is accompanied by the constrain that the whole information \( I_{sys} \) in the system should not be changed during the maximization process

\[
I_{sys}(c_1, \ldots, c_k) = I_0 = \text{const}. \tag{2.8}
\]

By these two conditions the relative maximum (2.7) of the multivariate function \( I_{out} \) is searched under the constrain of (2.8). The standard method to solve a problem like this is the method of Lagrange multipliers. For this purpose, let us define the differentiable function

\[
L(c_1, \ldots, c_k, \lambda) := I_{out}(c_1, \ldots, c_k) + \lambda I(c_1, \ldots, c_k) \quad \text{Lagrange function} \tag{2.9}
\]

with the constrain \( I(c_1, \ldots, c_k) := I_{sys}(c_1, \ldots, c_k) - I_0 = 0 \).

Since the Lagrange function includes the restriction, the necessary conditions for a relative maximum of the Lagrange function gives us the optimal values for the system parameters

\[
\frac{\partial L}{\partial c_1}(c_1^*) = 0, \ldots, \frac{\partial L}{\partial c_k}(c_k^*) = 0, \quad \frac{\partial L}{\partial \lambda}(\lambda^*) = 0 \tag{2.10}
\]

The conditions above transform to the equations

\[
\frac{\partial I_{out}(c_1^*)}{\partial c_1} + \lambda \frac{\partial I(c_1^*)}{\partial c_1} = 0, \ldots, \frac{\partial I_{out}(c_k^*)}{\partial c_k} + \lambda \frac{\partial I(c_k^*)}{\partial c_k} = 0, \tag{2.11a}
\]

\[
I(c_1^*, \ldots, c_k^*) = 0. \tag{2.11b}
\]

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

\[
\frac{\partial I(c_i)}{\partial c_i} = \frac{\partial I_{sys}(c_i)}{\partial c_i} = \left[ \frac{\partial c_i}{\partial I_{sys}(c_i)} \right]^{-1} \tag{2.12}
\]

and the conditions (2.11a,b) become

\[
\frac{\partial I_{out}(c_1^*)}{\partial c_1} \frac{\partial I_{sys}}{\partial I_{sys}} = -\lambda = \ldots = \frac{\partial I_{out}(c_k^*)}{\partial c_k} \frac{\partial I_{sys}}{\partial I_{sys}} \tag{2.13a}
\]
Eqs. (2.13a) say that for the necessary conditions of an optimal information distribution all the terms on the left hand side of (2.13a) should be equal: This is the principle of optimal information distribution as it is stated above in section 2.1 and expressed in equation (2.6). The last condition (2.13b) is just our well-known restriction (2.8).

3. Application Examples

In this section, first we want to demonstrate the above procedure 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 second section 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 [2].

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 Fig. 3.

Another version of the quadratic function is the logistic function \( x(t+1) = f(x) := ax(1-x) = ax-ax^2 \) which yields deterministic chaotic behavior in the interval \([0,1]\) for some values of \( a \) [3]. This system can be approximated by the network of Fig. 2, 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 [9].

Let us return to our example of the quadratic function \( f(x) = ax^2 + b \). Each neuron of the network of Fig. 3 has the output \( y_i \) with the output function \( y_i = S(z_i) \) and the activation function \( z_i \).
\[ z_i = \sum_j w_{ij} x_j , \]  
which becomes for the first layer

\[ z_i = w_i x + t_i \quad \text{with the threshold } t_i, \]  
and for the second layer

\[ \hat{f}(x) = \sum_i w_i S(z_i) + T \quad \text{with the threshold } T. \]  

Let us assume that we use a simple limited linear output function as squashing function

\[ S(z_i) = \begin{cases} 
1 & 1 < z_i , \\
z_i & 0 < z_i < 1 , \\
0 & z_i < 0 . 
\end{cases} \]  

The definition (3.4) satisfy the conditions \( S(\infty) = 1, S(-\infty) = 0 \) of [6] and is shown in Fig. 2 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 analog, linear electronic amplifier with output signal limits.

Let us assume that all the weights have converged by a proper learning 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 one of the others, the superposition will again yield only a line, resulting in a bad approximation of a parabola by one line. To obtain as many approximating lines as possible, the learning algorithm have 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 divide the whole input interval \([x_0,x_1]\) by the \( m \) neurons of the first layer into \( m \) equal (see app. B) 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 \( t_i \) of the previous intervals it represents the linear part of the approximation function \( \hat{f}(x) \) in the interval \([x_i-\Delta x/2, x_i+\Delta x/2]\):

\[ \hat{f}(x) = \sum_{i=1}^{k} W_i S(z_i) + T = \sum_{i=1}^{k} W_i + T + W_k S(z_k) . \]  

The resulting approximation is shown in Fig. 4.

The corresponding values for \( w_i, t_i, W_i \) and \( T \) can be easily analytically calculated. From the conditions of (3.4) we can conclude

\[ z_i|_{x_i-\Delta x/2} = 0 \quad \text{and} \quad z_i|_{x_i+\Delta x/2} = 1 \]

and by (3.2) we get

\[ w_i = 1/\Delta x = m/(x_1-x_0), \quad t_i = -w_i (x_i-\Delta x/2) = x_0/\Delta x +1-i = -mx_i/(x_1-x_0) +1/2 . \]  

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

\[ \frac{\partial \hat{f}(x_i)}{\partial x} = \frac{\partial \hat{f}(x_i)}{\partial x} \bigg|_{x_i} = 2ax_i := \Delta y/\Delta x . \]

Since the output \( S(z) \) is normalized between 0 and 1, we have to choose the weights \( W_i \)
as the normalized tangent at $\Delta x = 1$. Therefore, the weights become

$$W_i := \Delta y/1 = 2ax_i \Delta x . \tag{3.6c}$$

Then the basic threshold $T$ becomes the offset of the approximation at $x_0$, see Fig. 3. Using Eq.(B.1) we get

$$T = f(x_0) - d_{\text{lin}} = ax_0^2 + b - a/2 (\Delta x/2)^2 . \tag{3.6d}$$

**Example:**

For a net of $m=5$ neurons we get for $a=1$, $b=0$ with $\Delta x = 0.4$ five non-overlapping intervals $[-1,-0.6],[-0.6,-0.2],[-0.2,0.2],[0.2,0.6],[0.6,1.0]$ and $x_i = \{-0.8,-0.4,0.0,0.4,0.8\}$, $W_i = \{-0.64,-0.32,0.0,0.32,0.64\}$, $w_i = 2.5$, $t_i = \{+2.5, +1.5, +0.5, -0.5, -1.5\}$, $T=0.98$. The maximal approximation error $d_{\text{lin}} = 0.02$ has the same order as in the simulation results of Lapedes and Farber [9].
In Fig. 5 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 Fig. 4 (and Fig. B.1) we might suppose that the error of the approximation does not remain constant, but has minimal and maximal values. This is confirmed in Fig. 6 for the example of five neurons.

![Graph showing linear approximation error](image)

**Fig. 6.** The linear approximation error in the interval $x \in [-1, +1]$ for $m=5$ neurons.

In real-world applications we are not interested in the mean error over the interval (which is approximately zero in the above example), but in the maximal error that can occur. Thus, we aim not to minimize the average error of the approximation, but to minimize the **maximal** error. As the error of the linear approximation, we consider therefore the maximal linear approximation error $d_{\text{lin}}^{\text{max}}$ which is evaluated in appendix B to

$$d_{\text{lin}}^{\text{max}} = \frac{a}{2} \left(\frac{\Delta x}{2}\right)^2.$$  \hfill (B.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 information after (2.3) for $w_i$, $t_i$, $W_i$ and $T$, we have to define first the range $V_w, V_t, V_W$ and $V_T$ of the variables. For the sake of simplicity, let us assume that the value ranges and the information content of all variables are independant of the index $i$. Since the variables $w$ and $T$ are constant, they might be implemented in read-only-memory (ROM) with $\min(w_i) = 0 = \min(T)$ and thus by (3.6a,b,c,d), we have

\[
\begin{align*}
\max(w_i) - \min(w_i) &= V_w = \frac{m}{(x_1-x_0)} , \\
\max(t_i) - \min(t_i) &= V_t = [-mx_0/(x_1-x_0) + 1/2] - [-mx_0/(x_1-x_0) + 1/2] = m , \\
\max(W_i) - \min(W_i) &= V_W = 2a(x_1-x_0)\Delta x = 2a(x_1-x_0)^2/m , \\
\max(T) - \min(T) &= V_T = ax_0^2 + b - a/2 (\Delta x/2)^2.
\end{align*}
\]  \hfill (3.7a,b,c,d)
The maximal resolution error $\delta$ of a variable in one state is just the half of the resolution increment $d$ of Eq. (2.3)

$$\delta = d/2 = V/2 \cdot 2^{-1}$$  \hspace{1cm} (3.8a)

and therefore

$$\delta_w = V_w/2 \cdot 2^{-1w} = m/(x_1-x_0)2 \cdot 2^{-1w} , \hspace{1cm} (3.8b)$$

$$\delta_t = 1/2 m \cdot 2^{-1t} , \hspace{1cm} (3.8c)$$

$$\delta_w = a(x_1-x_0)^2/m \cdot 2^{-1w} , \hspace{1cm} (3.8d)$$

$$\delta_T = a/2 \left( x_0^2+b/a - 1/2 \left( (x_1-x_0)/(2m) \right)^2 \right) \cdot 2^{-1T} =: a/2 g_{T}(m) \cdot 2^{-1T} . \hspace{1cm} (3.8e)$$

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 C the error $d_{res}^{\text{max}}$ due to the finite resolutions $I_w, I_t, I_w, I_T$ and $m$ is evaluated to

$$d_{res}^{\text{max}} = 2a_1 \Delta x \left[ \delta_w T(x_1+\delta_t) + m\delta_w + \delta_T . \right. \hspace{1cm} (C.2)$$

### 3.3. The Optimal Information Distribution

As we have already mentioned, we are not interested in minimizing the average error of the approximation. Besides, since we do not assume anything about the input probability distribution $p(x)$, we cannot compute the average error. Instead, as a performance measure of the approximation network, let us compute the maximal possible error. The maximal approximation error is given by the worst case condition that the linear approximation error $d_{lin}$ and the resolution error $d_{res}$ do not compensate each other but adding up to

$$d_{l}^{\text{max}} = d_{lin}^{\text{max}} + d_{res}^{\text{max}} . \hspace{1cm} (3.9)$$

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

$$I_{sys} = m(I_w+I_t+I_w) + I_T . \hspace{1cm} (3.10)$$

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 Fig. 7 the approximation error is shown for different values of $m$ and constant system information $I_{sys}=708$ bits; the number of bits for all other variables are the same $I_w=I_t=I_{sys}=I_T$ and can be directly computed by Eq. (3.10).

The minimal error of $d_{l}^{\text{max}}=2.28x10^{-3}$ 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 multi-dimensional minimum of $d_{l}^{\text{max}}(m,I_w,I_t,I_w,I_T)$ which we have already solved in sections 2.1 and 2.2. The condition (2.6) for an optimal information distribution becomes
\[
\frac{\partial}{\partial m} \left( d_{\text{lin}}^{\text{max}} + d_{\text{res}}^{\text{max}} \right) \left( \frac{\partial I_{\text{sys}}}{\partial m} \right)^{-1} = \ldots = \frac{\partial}{\partial I_T} \left( d_{\text{lin}}^{\text{max}} + d_{\text{res}}^{\text{max}} \right) \left( \frac{\partial I_{\text{sys}}}{\partial I_T} \right)^{-1}, \quad (3.11)
\]

\[
\log(d^{\text{max}}) \quad 10^{-1}
\]

\[
\begin{align*}
I_{\text{sys}} &= 708.45 \text{ Bit} \\
2 & 4 6 8 10 12 14 16 18 20 22 24 26 28 30 m
\end{align*}
\]

Fig. 7. The approximation error at constant system information (a=1, b=0).

with the derivatives of (3.10)
\[
\begin{align*}
\frac{\partial I_{\text{sys}}}{\partial m} &= I_w + I_T + I_W \\
\frac{\partial I_{\text{sys}}}{\partial I_w} &= m = \frac{\partial I_{\text{sys}}}{\partial I_T} = \frac{\partial I_{\text{sys}}}{\partial I_W} = 1. \quad (3.12)
\end{align*}
\]

The five terms of (3.11) should all be equal, giving us four equations with five variables. In app. D, this is evaluated giving us the three equations
\[
\begin{align*}
I_T &= I_w + C, \quad \text{with } C := \log((x_1 - x_0)/x_1), \quad (D.4) \\
I_W &= I_T + C, \quad (D.8) \\
I_T &= I_W + \log(g_T(m)/2) - \log((x_1 - x_0)^2/m), \quad (D.6)
\end{align*}
\]

and the equation for the number of neurons
\[
m = h(m, I_T)^{1/3}. \quad (D.11)
\]

This we can use for numerically given $I_T$ as an iteration formula at the $(t+1)^{\text{th}}$ iteration for $m$:
\[
m(t+1) = h(m(t), I_T)^{1/3}. \quad (3.13)
\]

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

**Example**

Let us consider an information of 16 bits in the threshold $T$. In the simple case of $x_0=-1, x_1=+1, a=1, b=0$ we have with $I_T:=16$ bit, $C=1$ bit the optimal configuration at $m = 16.54$ neurons, $I_W = 14.95$ bit, $I_T=I_W-C = 13.95$ bit, $I_w = I_T-C = 12.95$ bit.

The overall information in the network is then with (3.10) $I_{\text{sys}} = m(I_w+I_T+I_W) + I_T = 708.45$ [bits] and the approximation error is $d_T^{\text{max}} = 2.213\times10^{-3}$. If we augment the information capacity of the system to $I_T=32$ Bit, the error will diminish to $d_T^{\text{max}} = 1.847\times10^{-6}$ when we use the optimal system parameters.
Fig. 8 shows the optimal system parameter $m$ when one parameter (the threshold information $I_T$) is given at $a=1$, $b=0$, $x_0=-1$, $x_1=1$. The corresponding values for $I_W$ and the overall system information $I_{sys}$ are also plotted. Since the values for $I_t$ and $I_w$ differ from $I_W$ only by a constant offset of one and two bits, they are omitted in the figure for clarity.

In Fig. 9 the minimal approximation error for optimal system parameters is shown in logarithmic notation for the whole interval of $I_T = 4 .. 32$ bits. The nearly linear appearance is due to the fact that all terms of the resolution error contains powers of two, which transforms to linear terms in $I_T$.

The corresponding approximation error for a partially optimal information distribution with equal resolutions $I_W = I_t = I_{W} = I_T$, but balanced to the number of neurons $m$, are generally slightly worse than the one for an optimal information distribution.
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 \textit{kinematic} control computes the Cartesian position of the endpoint of a robot manipulator, composed of several segments and joints, by a straightforward matrix multiplication (\textit{homogeneous transformation}) of all segment-matrices when the joint coordinates (joint angles) are given. The inverse transformation, the \textit{inverse kinematics}, does the inverse task: when the absolute Cartesian coordinates $x$ of the endpoint (e.g. the palm of the robot hand) is given, it computes the appropriate joint coordinates $\theta_i$ for each segment.

The inverse kinematic of a robot is a quite complicated function and not easy to find. Furthermore, when the rotation 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 neighborhood-conserving mappings [8] by Ritter, Martinetz and Schulten [12]. Since the mapping is very raw for a small amount of neurons, they additionally use a linear approximation with learned coefficients. In Fig. 10 the neural network for the robot control is shown.

![Fig. 10. The approximator network for robot control.](image)

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. Here we have to balance the number $n$ of storage cells (number of neurons) per dimension against the bits per cell (resolutions $I_w$, $I_g$, $I_A$ of the weights and coefficients). The choice for the system parameters $n$, $I_w$, $I_g$, $I_A$ 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 rests an error $d_{\text{lin}}$ due to the discrete approximation of the non-linear function. For the example of the commonly used PUMA robot (Fig. 11), this was evaluated in [3], based on the strategy for optimal storage distribution, studied in [2]. The main results are given below.

![Fig. 11. The PUMA robot (after [FU87]).](image)

Let us first evaluate the error $d_{\text{lin}}$ due to the linear approximation. Since we have only rotational axes in the system, the most difficult task 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.

![Fig. 12. The absolute positioning error as a function of $n$ (neurons per dim).](image)
The numerically computed approximation error is shown in Fig. 12. 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 Fig. 13 the logarithm of the joint error is drawn versus the number \( n \) of the neurons.

![Fig. 13. The joint error as a function of \( n \).](image)

This gives us the analytical expression of \( d_{\text{lin}}^{\text{max}} = B n^b \) as a good approximation with numerically obtained values for \( B \) and \( b \). This coincides well with the analytical expression (B.3) for the linear approximation error of the example of a quadratic function.

The resolution error \( d_{\text{res}}^{\text{max}} \) of the linear approximation scheme can be straightforwardly calculated by the same ideas as for equation (C.2).

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

\[
d_{\text{max}} = |d_{\text{lin}}^{\text{max}} + d_{\text{res}}^{\text{max}}|.
\]

Since the form of both errors are now analytically known, the conditions for the optimal information distribution of Eq. (2.4) can be calculated, using the derivatives of \( d_{\text{max}}^{\text{max}} \), i.e. of \( d_{\text{lin}}^{\text{max}} \) and of \( d_{\text{res}}^{\text{max}} \). 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 Fig. 14.

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 Fig. 15 for the situation when all weights and thresholds are forced to have the same resolution (number of bits per variable) but optimal \( n \) and, additionally, when they all have different, optimal resolutions.
Fig. 14. The optimal parameter configurations for minimal information storage.

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 with the resolution of $I_w=16.4$ Bits for all weights and coefficients. The optimal configuration with different resolutions gives only a 18% smaller error, and therefore do not encourage the use of multiplication operations with variable accuracy which will be necessary in this case.

Fig. 15. The Cartesian error at minimal system information.

4. Conclusion

The principle of optimal information distribution is a criterion 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 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.

Nevertheless, for future work it remains to find procedures more efficient than the general backpropagation or the Kohonen map for the training of the approximation layers.

References


Appendix A: The optimal output classes

The output information is maximized when
\[ H(y) = \langle l_{\text{out}} \rangle = \max \]

With the notation \( P_i := P(y_i) \) the condition becomes
\[ H(P_1, \ldots, P_N) := -\sum_i P_i \ln P_i \leq \max \]
with the constrain \( \sum_i P_i = 1 \) or \( g(P_1, \ldots, P_N) := \sum_i P_i - 1 = 0 \).

The maximum condition and the restriction are satisfied when the Lagrange function
\[ L(P_1, \ldots, P_N, \lambda) := H(P_1, \ldots, P_N) + \lambda g(P_1, \ldots, P_N) \]
becomes maximal. The necessary conditions for this are

(A.1a) \[ \frac{\partial}{\partial P_i} L(P_1^*, \ldots, P_N^*) = 0 \]
(A.1b) \[ \frac{\partial}{\partial \lambda} L(\lambda^*) = 0 \]

From (A.1a) we get for each \( P_i \)
\[ \frac{\partial}{\partial P_i} L(P_1^*) = \frac{\partial}{\partial P_i} H(P_1^*, \ldots, P_N^*) + \lambda \frac{\partial}{\partial P_i} g(P_1^*, \ldots, P_N^*) = -[\ln (P_i^*)+1] + \lambda = 0 \]
and from (A.1b) our constrain \( g(.)=0 \). Since we get for two arbitrary indices \( i \) and \( j \) the equation
\[ P_i^* = \exp(\lambda-1) = \text{const} = P_j^* \]
the probabilities are all equal and we can conclude with the restriction \( \sum P_i = 1 \) that
\[ P(y_i)^* = P(y_j)^* = 1/N \quad \text{for all } i,j \]

Appendix B: The linear approximation error

The non-linear function in the interval \([x-\Delta x/2, x+\Delta x/2]\) is
\[ f(x) = ax^2 + b \]
and the linear approximation by the neural network is
\[ \hat{f}(x) = \alpha x + \beta \quad \text{with } \alpha := 2ax \]

The approximation error is (see Fig. 3 and B.1)
\[ d_{\text{lin}}(x) = f(x) - \hat{f}(x) = ax^2 + b - 2axx - \beta = b - \beta - ax^2 =: d, \]
\[ d_{\text{lin}}(x+\Delta x/2) = f(x+\Delta x/2) - \hat{f}(x+\Delta x/2) = d + a(\Delta x/2)^2, \]
\[ d_{\text{lin}}(x-\Delta x/2) = f(x-\Delta x/2) - \hat{f}(x-\Delta x/2) = d + a(\Delta x/2)^2. \]

Thus, the errors at the boarders are all equal.
The maximal error \( \max(|d_{\text{lin}}(x)|,|d_{\text{lin}}(x+\Delta x/2)|) \) is minimal when all the errors are equal
\[ |d_{\text{lin}}(x)| = |d_{\text{lin}}(x+\Delta x/2)|, \]
or
\[ d_{\text{lin}} = d + a(\Delta x/2)^2. \]
This is given when
\[ d := -a(\Delta x/2)^2/2. \]
The maximal linear error is not dependant on the value of \( x \), it is the same in the whole interval
\[ d_{\text{lin}}^{\max} = a(\Delta x/2)^2/2. \] (B.1)
Since we have \( \Delta x = (x_1-x_0)/m \),
\[ d_{\text{lin}}^{\max} = a((x_1-x_0)/2m)^2/2 = m^{-2}a(x_1-x_0)^2/8, \] (B.2)
and therefore
\[ d_{\text{lin}}^{\max} = C m^b \quad \text{with } C := a(x_1-x_0)^2/8 \quad \text{and } b := -2. \] (B.3)

Appendix C: 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 independant of its index, i.e. all weights in one layer are assumed to have equal resolution. Then, the approximating function becomes with (3.2) and (3.3)
\[
\tilde{f}(x,\delta) = \sum_i (W_i + \delta_W)S(z_i+\delta_z) + T + \delta_T = \sum_i W_i S(z_i+\delta_z) + T + \sum_i \delta_W S(z_i+\delta_z) + \delta_T
\] (C.1)
Because the intervals are exclusive, for the \( k \)th interval, 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 \( S(z_i+\delta_z) = 0 \).
\[
\tilde{f}(x,\delta) = (\sum_i^{k-1} W_i) + W_k S(z_k+\delta_z) + T + (\sum_i^{k-1} \delta_W) + \delta_W S(z_k+\delta_z) + \delta_T
\] = \( \tilde{f}(x) + W_k \delta_z + (k-1)\delta_W + \delta_W S(z_k+\delta_z) + \delta_T \).
The maximal error \( d_{\text{res}}^{\text{max}} \) is encountered at \( \max(x) = x_1 \) on the boarder of the interval \([x_0, x_1]\). The contribution of the term \( \delta_w S(.) \) becomes maximal \( \delta_w \) when \( S(.) = 1 \). Therefore, we have

\[
\hat{f}(x_1, \delta) = f(x_1) + (m-1)\delta_w + W_m \delta z_m + \delta_w S(z_m+\delta_z) + \delta_T
\]

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

\[
d_{\text{res}}^{\text{max}} = \hat{f}(x_1, \delta) - f(x_1) = m\delta_w + W_m (\delta_w x_1 + \delta_t) + \delta_T.
\]

With (3.6c), we get

\[
d_{\text{res}}^{\text{max}} = 2ax_1 \Delta x [\delta_w x_1 + \delta_t] + m\delta_w + \delta_T.
\]

Using the definitions (3.8b,c,d,e) we get

\[
d_{\text{res}}^{\text{max}}(m) = 2ax_1(x_0-x_1)\left[ \frac{x_1}{2} + \frac{1}{2} \right] + \frac{a_1}{2} (x_1-x_0)^2 + \frac{1}{2} [ax_0^2 + b - a_1 (x_1-x_0)^2] 2^{-3T}.
\]

**Appendix D: The evaluation of the optimal information distribution parameters**

Let us evaluate the derivative of the first parameter in (3.11). With (B.2) we have

\[
\frac{\partial}{\partial m} d_{\text{lin}}^{\text{max}} = \frac{\partial}{\partial m} \frac{a}{\delta} (x_1-x_0)^2 m^{-2} = -\frac{a}{4m^3} (x_1-x_0)^2 (2^{-T} m^{-2}).
\]

and with (C.3) we have

\[
\frac{\partial}{\partial m} d_{\text{res}}^{\text{max}} = \frac{a}{8} (x_1-x_0)^2 2^{-T} m^{-2}.
\]

Therefore, the expressions (D.1) and (D.2) together with (3.12) yields the first term of the equations in (3.11),

(i) \[
\frac{\partial}{\partial m} d_{\text{lin}}^{\text{max}} \left( \frac{\partial L_{\text{sys}}}{\partial m} \right)^{-1} = -\frac{a}{4m^3} (x_1-x_0)^2 \left[ 1 - 2^{-T} m^{-2} \right] (I_w + I_t + I_w)^{-1}.
\]

All the other system parameters \( I_w, I_t, I_w, I_T \) do not influence the linear approximation error \( d_{\text{lin}}^{\text{max}} \). Therefore, the derivation of the error (B.2) is zero and we get the terms

(ii) \[
\frac{\partial}{\partial I_w} d_{\text{lin}}^{\text{max}} \left( \frac{\partial L_{\text{sys}}}{\partial I_w} \right)^{-1} = 2ax_1 \Delta x \frac{\partial \delta_w}{\partial I_w} m^{-1} = -2x_1^2 a (x_1-x_0) \ln(2) \delta_w m^2
\]

because \( (\delta_w)^{-1} \frac{\partial \delta_w}{\partial I_w} = \frac{\partial}{\partial I_w} \ln(\delta_w) = \frac{\partial}{\partial I_w} (\ln(V_w) - I_w \ln(2)) = -\ln(2) \)

(iii) \[
\frac{\partial}{\partial I_t} d_{\text{lin}}^{\text{max}} \left( \frac{\partial L_{\text{sys}}}{\partial I_t} \right)^{-1} = 2ax_1 \Delta x \frac{\partial \delta_t}{\partial I_t} m^{-1} = -2x_1 a \delta_w (x_1-x_0) \ln(2) \delta_t
\]

(iv) \[
\frac{\partial}{\partial I_w} d_{\text{lin}}^{\text{max}} \left( \frac{\partial L_{\text{sys}}}{\partial I_w} \right)^{-1} = \frac{\partial m \delta_w}{\partial I_w} m^{-1} = -\ln(2) \delta_w
\]

(v) \[
\frac{\partial}{\partial I_T} d_{\text{lin}}^{\text{max}} \left( \frac{\partial L_{\text{sys}}}{\partial I_T} \right)^{-1} = \frac{\partial \delta_T}{\partial I_T} m^{-1} = -\ln(2) \delta_T
\]
All the five terms should be equal to yield an optimal information distribution. Let us evaluate the equalities.

**With term (ii) = term (iii), we know that**

\[ x_1 \delta_W = \delta_t \]  \hspace{1cm} (D.3)

The resolution errors of the weights and the threshold of the first layer should be in the same order since they produce the same final error by multiplication with \( W \).

The equation (D.3) gives us with (3.8b) and (3.8c),

\[
\begin{align*}
\frac{x_1}{x_1-x_0} \cdot 2^{I_W} &= m \cdot 2^{-I_t} , \\
\text{ld}(2^{I_t}) &= \text{ld}((x_1-x_0)/x_1) + \text{ld}(2^{I_W}) , \\
I_t &= I_W + C \quad \text{with } C := \text{ld}((x_1-x_0)/x_1). \hspace{1cm} (D.4)
\end{align*}
\]

The information of the threshold has a constant offset from the information of the weights. For the case of \( x_0 = -1, x_1 = +1 \), we have with \( C = 1 \) just one bit offset.

**term (iv) = term (v)**

The corresponding case for the threshold and weights of the second layer reveals

\[ \delta_{W} = \delta_{T} . \]  \hspace{1cm} (D.5)

The threshold should be as fine grained as the weights since it is always involved in the output accuracy. Eq. (D.5) gives with (3.8d) and (3.8e),

\[
\begin{align*}
a(x_1-x_0)^2/m & \cdot 2^{-I_W} = a/2 \cdot g_T(m) \cdot 2^{-I_T} , \\
\text{and therefore} \\
\text{ld}(2^{I_T}) &= \text{ld}(2^{I_W}) + \text{ld}(g_T(m)/2) - \text{ld}((x_1-x_0)^2/m), \\
I_T &= I_W + \text{ld}(g_T(m)/2) - \text{ld}((x_1-x_0)^2/m) . \hspace{1cm} (D.6)
\end{align*}
\]

The threshold information of the second layer has also an offset to the weights and depends on the number of inputs from the first layer.

**term (iii) = term (iv)**

The comparison between the threshold of the first layer and the weights of the second layer gives

\[ \frac{2a}{m} \cdot x_1(x_1-x_0) \delta_t = \delta_W \]  \hspace{1cm} (D.7)

and therefore using (3.8c) and (3.8d)

\[
\begin{align*}
\frac{a}{m} \cdot x_1(x_1-x_0) \cdot 2^{I_t} &= \frac{a}{m} \cdot (x_1-x_0)^2 \cdot 2^{I_W} , \\
\text{ld}(2^{I_t}) &= \text{ld}(2^{I_W}) + \text{ld}(x_1/(x_1-x_0)) , \\
I_t &= I_W - C . \hspace{1cm} (D.8)
\end{align*}
\]
term (i) = term (v)

The condition for the number of neurons is

\[ \frac{a}{4m^3} (x_1 - x_0)^2 (1 - 2^{-1/T}/2) (I_w + I + I_w)^{-1} = \ln(2) \delta_T . \] (D.9)

Using Eqs. (D.4), (D.8) and (3.8e) the condition (D.9) becomes

\[ \frac{a}{4m^3} (x_1 - x_0)^2 (1 - 2^{-1/T}/2) = 3 (I_w - C) \ln(2) a/2 g_T(m) 2^{-1/T} , \]

\[ (x_1 - x_0)^2 (2^{1/T} - 1/2) = 6m^3 (I_w - C) \ln(2) g_T(m) , \]

and finally using Eq. (D.6)

\[ 6m^3 (I_T - \ln(g_T(m)/2) + \ln((x_1 - x_0)^2/m) - C) \ln(2) g_T(m) - (x_1 - x_0)^2(2^{1/T} - 1/2) = 0 , \]

which can be put into the form

\[ m = h(m,I_T)^{1/3} . \] (D.11)

Since an analytical solution for this equation is not so easy to obtain, we can compute the desired optimal value \( m^* \) as the fixpoint of a numerical iteration for given values of the other parameters \( I_T, a, x_1, x_0 \).