Perturbation Method for Deleting Redundant Inputs of Perceptron Networks*

Jacek M. Zurada†, Aleksander Malinowski†, Shiro Usui‡

†Department of Electrical Engineering
University of Louisville, Louisville, KY 40292, USA
e–mail: jmzura02@starbase.spd.louisville.edu

‡Toyohashi University of Technology, Toyohashi, Japan
e–mail: usui@bpel.tutics.tut.ac.jp

ABSTRACT: Multilayer feedforward networks are often used for modeling complex functional relationships between data sets. Should a measurable redundancy in training data exist, deleting unimportant data components in the training sets could lead to smallest networks due to reduced–size data vectors. This reduction can be achieved by analyzing the total disturbance of network outputs due to perturbed inputs. The search for redundant input data components proposed in the paper is based on the concept of sensitivity in linearized models. The mappings considered are $\mathbb{R}^{I} \rightarrow \mathbb{R}^{K}$ with continuous and differentiable outputs. Criteria and algorithm for inputs' pruning are formulated and illustrated with examples.

INTRODUCTION

Neural networks are often used to model complex functional relationships between sets of experimental data. Such modeling approach proves useful when analytical models of processes do not exist or are not known, but when sufficient data is available for embedding existing relationships into neural network structures. Multilayer feedforward neural networks (MFNN) consisting of continuous neurons have been found particularly useful for such model building [1–3]. Representative training data are used in such case for supervised training of a suitable user–selected MFNN architecture.

Minimization of potential redundancy in data used for supervised training can take different forms. Duplicative data pairs are essentially removable from the training sets without a loss of accuracy. In contrast, special attention should be paid to data that carry conflicting information. Such data do not normally allow for unique mapping and should be eliminated. Our concern in this paper is

*This work was supported in part by the ONR Grant N00014–93–1–0855

to explore potential redundancy in input vector dimensionality. As such, this concern has only little in common with widely used notion of network pruning. By deleting superfluous inputs, if such inputs exist, the number of input nodes is reduced. The resulting network is still pruned as it contains no weights fanning out of deleted inputs.

A popular objective of network pruning is to detect irrelevant weights and neurons. This can be achieved through evaluation of sensitivities of the error function to the weights which are the learning parameters [5–6]. Errors other than quadratic are often used to achieve identification of insensitive weights. Statistical moments of neural networks–built mappings, including sensitivities to inputs, are discussed in [7]. Our focus in the paper is mainly to develop clear and practical measures of sensitivities to inputs rather than to weights or neurons. Then, a systematic algorithmic approach has been developed to utilize these measures towards deletion of redundant inputs.

To determine which inputs are necessary for the satisfactory neural network performance a metric known as saliency was introduced in [8]. Belue and Bauer developed an algorithm extending the saliency metric over the entire input space [9]. The approach involves multiple neural network training and superposition of noise on the training patterns to reduce the dependence of results on local minima. This method, however, is computationally intensive due to the required multiple training sessions and exhaustive coverage of the input space.

The saliency method was developed to determine the irrelevant features for neural network classifiers [9]. The sensitivities of MFNN outputs with respect to inputs are calculated and used along with various metrics to evaluate importance of features. Such classifier networks in general are characterized by small sensitivities, when fully trained. Therefore saliency can be applied only with the addition of noise to the training patterns and with sampling of the input space over the whole domain. Multiple training is necessary to average the results and prevent dependence on local minima achieved during training.

This paper focuses on the concept of sensitivity, or perturbation method, for pruning unimportant inputs for neural networks providing continuous mapping. This assumption and the proposed new

sensitivity summation metrics allow application of the method directly to trained MFNNs without adding noise to the training patterns or multiple trainings. In fact, in case of continuous mapping the problem of local minima reached during training is not important if sufficient approximation accuracy is achieved. As a result the presence of local minima does not affect the Jacobian matrix used by this method. The Jacobian matrix is derived from the approximate neural network mapping over the training data set. This eliminates the need for computationally intensive repetitive training. In addition to mappings with continuous outputs the sensitivity method can be applied to the classification problems. However, in such cases an additional neural network has to be trained as described in one of the examples.

Let us consider an MFNN with a single hidden layer. The network is assumed to perform a nonlinear, differentiable mapping Γ: $\mathbb{R}^{I} \rightarrow \mathbb{R}^{K}$, $\mathbf{o}=\Gamma(\mathbf{x})$, where $\mathbf{o}$ (Kx1), and $\mathbf{x}$(Ix1) are output and input vectors, respectively. In further discussion it is assumed that certain inputs bear none, or little statistical or deterministic relationships to output vectors, and are therefore removable. The objective here is to reduce the original dimensionality of the input vector, $\mathbf{x}$, so that a smaller network can be used as a model without loss of accuracy. Initial considerations published in [10–12] are extended below along with a formal framework for the perturbation approach as applied to the neural network models.

Let $\mathbf{o}$: $\mathbb{R}^{I} \rightarrow \mathbb{R}^{K}$ with component functions $o_1, o_2, ..., o_K$. Suppose $\mathbf{x}^{(n)} \in \Omega$, where Ω is an open set. Since $\mathbf{o}$ is differentiable at $\mathbf{x}^{(n)}$ we have

$$\boldsymbol{o}(\boldsymbol{x} + \Delta\boldsymbol{x}) = \boldsymbol{o}\left(\boldsymbol{x}^{(n)}\right) + \boldsymbol{J}\left(\boldsymbol{x}^{(n)}\right)\Delta\boldsymbol{x} + \boldsymbol{g}(\Delta\boldsymbol{x}) \tag{1}$$

where

$$\boldsymbol{J}\left(\boldsymbol{x}^{(n)}\right) = \begin{bmatrix} \frac{\partial o_1}{\partial x_1} & \frac{\partial o_1}{\partial x_2} & \cdots & \frac{\partial o_1}{\partial x_I} \\ \frac{\partial o_2}{\partial x_1} & \frac{\partial o_2}{\partial x_2} & \cdots & \frac{\partial o_2}{\partial x_I} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial o_K}{\partial x_1} & \frac{\partial o_K}{\partial x_2} & \cdots & \frac{\partial o_K}{\partial x_I} \end{bmatrix}_{\boldsymbol{x}^{(n)}}$$

is the Jacobian matrix and

$$\lim_{\Delta x \to 0} \frac{\boldsymbol{g}(\Delta \boldsymbol{x})}{|\Delta \boldsymbol{x}|} = \mathbf{0} \tag{2}$$

Fig. 1 provides geometrical interpretation of relationship (1) in space $\mathbb{R}^K$. Point $\mathbf{o}(\mathbf{x}^{(n)})$ represents the nominal response of the MFNN for the n–th element of the training set, $\mathbf{x}^{(n)}$. The disturbance $\Delta\mathbf{x}$ of the input vector causes the perturbed response at $\mathbf{o}(\mathbf{x}^{(n)}+\Delta\mathbf{x})$. This response can be expressed as a combination of three vectors as indicated in (1).

Assuming now that the network with input $\mathbf{x}$ is perturbed by applying $\Delta\mathbf{x}\rightarrow\mathbf{0}$, then the only relevant component in (1) becomes $\mathbf{J}(\mathbf{x}^{(n)})\Delta\mathbf{x}$ due to the fact that the first term of (1) is fixed, and the third one vanishes accordingly to (2). This corresponds to the shaded triangle vanishing due to the vanishing $\mathbf{g}(\Delta\mathbf{x})$ side, but also due to the vanishing multiplier $\Delta\mathbf{x}$ of the Jacobian matrix. Still, matrix $\mathbf{J}(\mathbf{x}^{(n)})$ provides the crucial first–order directional information about the non-zero displacement $\mathbf{o}(\mathbf{x}^{(n)}+\Delta\mathbf{x})–\mathbf{o}(\mathbf{x}^{(n)})$.

The proposed input perturbation approach has proven rather useful for function approximation cases studied in context of input pruning. However, in case of pattern classifiers output neurons are near saturation and the method needs to be modified as discussed in one of the later sections.

STATEMENT OF THE PROBLEM

Our purpose is to evaluate the displacements due to the perturbed inputs over the entire training set $\mathfrak{X}=\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(N)}\}$. For an example of several training vectors $\mathbf{x}^{(n)}\in\mathfrak{X}$ disturbed by vector $\Delta\mathbf{x}$ each output relationships are depicted on Fig. 2a. Depicted displacements are for identical and small values of $\Delta\mathbf{x}$ for i=1, 2, ... N (N=18 in this example).

These changes can be projected back to the input space $\mathbb{R}^I$. The question is whether or not all I dimensions of input vectors are relevant for having caused the displacements of outputs. Fig. 2b illustrates an example of respective input changes which are causing output perturbations of Fig. 2a. It can be seen that the variable x_2 does not participate in output changes $\mathbf{o}^{(k)}(\mathbf{x}^{(i)}+\Delta\mathbf{x})$, or each of the K output functions measured on the training set $\mathfrak{X}$ is constant in x_2.

In general, should all outputs be insensitive to the i–th variable, then the entire i–th column of the Jacobian matrix would vanish. Note that the vanishing column property would have to hold for the entire training set $\mathcal{X}$, thus zeroing the i–th column for $\mathbf{J}(\mathbf{x}^{(n)})$, n=1, 2, ..., N. In real life situations, however, qualitative measures other than zero need to be developed to compare the relative significance of each particular input over the training set. Following sections of the paper are aimed at formulating such measures and the related input variable pruning algorithm.

SENSITIVITY MATRIX

It can be easily noticed that the entries of the Jacobian matrix defined in (1) can be considered as sensitivity coefficients. Specially, sensitivity of an output o_k with respect to its input x_i is

$$S_{x_i}^{o_k} \doteq \frac{\partial o_k}{\partial x_i} \tag{3}$$

which can be written succinctly as

$$S_{ki} \doteq S_{x_i}^{o_k}$$

By using the standard notation of an error backpropagation approach [3], the derivative of (3) can be readily expressed in terms of network weights as follows

$$\frac{\partial o_k}{\partial x_i} = o_k' \sum_{j=1}^{J-1} w_{kj} \frac{\partial y_j}{\partial x_i} \tag{4}$$

where y_j denotes the output of the j–th neuron of the hidden layer, and o_k' is the value of derivative of the activation function o=f(net) taken at the k–th output neuron. This further yields

$$\frac{\partial o_k}{\partial x_i} = o_k' \sum_{j=1}^{J-1} w_{kj} y_j' v_{ji} \tag{5}$$

where y_j' is the value of derivative of the activation function y=f(net) of the j–th hidden neuron (y_J'=0 since the J–th neuron is a dummy one, i.e. it serves as a bias input to the output layer). The sensitivity matrix **S** (KxI) consisting of entries as in (5) or (3) can now be expressed using array notation as

$$\boldsymbol{S} = \mathbf{O}' \times \mathbf{W} \times \mathbf{Y}' \times \mathbf{V} \tag{6}$$

W (KxJ) and **V** (JxI) are output and hidden layer weight matrices, respectively, and **O'** (KxK) and **Y'** (JxJ) are diagonal matrices defined as follows

$$\begin{aligned} \boldsymbol{O}' &\doteq diag(o_1', o_2', ..., o_K') \\ \boldsymbol{Y}' &\doteq diag(y_1', y_2', ..., y_J') \end{aligned} \tag{7}$$

Matrix **S** contains entries S_{ki} which are ratios of absolute increments of output k due to the input i as defined in (3). This matrix depends only upon the network weights as well as slopes of the activation functions of all neurons. Each training vector $\mathbf{x}^{(n)} \in \mathcal{X}$ produces different sensitivity matrix $\mathbf{S}^{(n)}$ even for a fixed network. This is due to the fact that although weights of a trained network remain constant, the activation values of neurons change across the set of training vectors $\mathbf{x}^{(n)}$, n=1, 2, ..., N. This, in turn, produces different diagonal matrices of derivatives **O'** and **Y'**, which strongly depend upon the neurons' operating points determined by their activation values. These matrices contain linearized activation functions at their quiescent points.

MEASURES OF SENSITIVITY TO INPUTS OVER TRAINING SET

In order to evaluate the option of dimensionality reduction of input vectors, the sensitivity matrix as in (6) needs to be evaluated over the entire training set $\mathcal{X}$. Let us define the sensitivity matrix for the pattern x_n as $\mathbf{S}^{(n)}$. There are several ways to define the overall sensitivity matrix, each relating to the different objective function which needs to be minimized.

The *mean square average (MSA) sensitivities*, $S_{ki,\,avg}$, over the set $\mathcal{X}$ can be computed as

$$S_{ki,\,avg} \doteq \sqrt{\frac{\sum_{n=1}^{N} \left(S_{ki}^{(n)}\right)^2}{N}} \tag{8}$$

Matrix $\mathbf{S}_{avg}$ (KxI) is defined as $[\mathbf{S}_{avg}]=S_{ki,avg}$. This method of sensitivity averaging is coherent with the goal of network training which minimizes the mean square error over all outputs and all patterns in the training set.

The *absolute value average sensitivities*, $S_{ki,\,abs}$, over the set $\mathfrak{X}$ can be computed as

$$S_{ki,\,abs} \doteq \sqrt{\frac{\sum_{n=1}^{N} |S_{ki}^{(n)}|}{N}} \tag{9}$$

Matrix $\mathbf{S}_{abs}$ (KxI) is defined as $[\mathbf{S}_{abs}]=S_{ki,abs}$. Note that summing sensitivities across the training set requires taking their absolute values due to the possibility of cancelations of their taking negative and positive values. This method of averaging may be more advantageous than (8) when sensitivities $S_{ki}^{(n)}$, n=1, ..., N, are of disparate values.

The *maximum sensitivities*, $S_{ki,\,max}$, over the set $\mathfrak{X}$ can be computed as

$$S_{ki,\,max} \doteq \max_{n=1..N} \left\{ S_{ki}^{(n)} \right\} \tag{10}$$

Matrix $\mathbf{S}_{max}$ (KxI) is defined as $[\mathbf{S}_{max}]=S_{ki,max}$. This sensitivity definition allows to prevent pruning inputs which are rather relevant for the network, but relevance occurs only in a small percentage of input vectors of the whole training set. Infrequent but relevant relationships in the training set are masked due to the averaging in (8)–(9), but remain distinguishable for the measure (10). The drawback of this measure is that the significance of inputs can be overestimated and some unimportant inputs may remain after shrinking the input vector.

Any of the sensitivity measure matrices proposed in (8)–(10) can provide useful information as to the relative significance of each of the inputs in $\mathfrak{X}$ to each of the outputs. For the sake of brevity, however, mainly the MSA sensitivity matrix defined in (8) will be used in further discussion but other sensitivity measures will be used for comparison purposes. The cumulative statistical information resulting from (8) will be used along with criteria for reducing the number of inputs to the smallest number of them sufficient for accurate learning. These criteria are formulated in the next section.

Other useful measure of sensitivity used for the evaluation of input saliency was introduced in [9]. Instead of summarizing S_{ki} over the data set as in (9), the set of points $\mathfrak{X}$ created by uniform sampling of the input space $\mathfrak{D} \subset \mathbb{R}^I$ is used

$$S_{ki,\,sal} \doteq \frac{\sum_{\mathcal{G}} |S_{ki}^{(n)}|}{N_{\mathcal{G}}} \tag{11}$$

$N_{\mathcal{G}}$ is the number of samples in $\mathcal{G}$. The saliency measure $[\mathbf{S}_{sal}]=S_{ki,sal}$ allows for better estimation of the sensitivity over the entire input space. However, in highly dimensional space, and when $\mathcal{D}$ does not have the shape of hypercube, the summation (11) may be difficult to perform due to the problems with generating the set $\mathcal{G}$. It would be computationally intensive to sample uniformly high-dimensional hypercube. Furthermore, training patterns in $\mathcal{D}$ may not cover the domain uniformly and/or some of the samples generated may not represent the desired properties of the network. Our proposed measures do not suffer from these potential limitations.

CRITERIA FOR PRUNING INPUTS

Inspection of the MSA sensitivity matrix $\mathbf{S}_{avg}$ allows to determine which inputs affect outputs least. A small value of $S_{ki,avg}$ in comparison to others means that for the particular k–th output of the network, the i–th input does not significantly contribute per average to output k, and therefore could be possibly disregarded. This reasoning and results of experiments allow to formulate the following practical rule: *The sensitivity matrices for a trained neural network can be evaluated for both training and testing data sets; the norms of MSA sensitivity matrix columns can be used for ranking inputs according to their significance and for reducing the size of network accordingly through pruning less relevant inputs.*

When one or more of the inputs have relatively small sensitivity in comparison to others, the dimension of neural network can be reduced by removing them, and a smaller–size neural network can be successfully retrained in most cases. The criterion used in this paper for an algorithm determining which inputs can be removed is based on the so called largest gap method.

Suppose two inputs are providing important data for neural network. One of them has much larger relative change than the other one. In such case the sensitivity of the second output would be much larger than the first one due to the necessity of an additional amplification of the input by network weights. In the extreme the first of those two inputs may even be selected for pruning. To prevent

such cases additional scaling of matrices defined in (8)–(10) is necessary or additional data preprocessing is another solution. In fact the latter seems to be better due to the fact that it prevents hidden layer neuron saturation at the beginning of the training when all their weights remain random, and therefore speed up the training. Formulas proposed in (12) allow to scale inputs into the range of [–1;1]. They were used in examples presented in this paper.

$$\hat{x}_i^{(m)} \doteq \frac{x_i^{(m)} - \frac{\left(\max\limits_{n=1..N}\left\{x_i^{(n)}\right\} + \min\limits_{n=1..N}\left\{x_i^{(n)}\right\}\right)}{2}}{\left(\max\limits_{n=1..N}\left\{x_i^{(n)}\right\} - \min\limits_{n=1..N}\left\{x_i^{(n)}\right\}\right)}$$

$$\hat{o}_k^{(m)} \doteq \frac{o_k^{(m)} - \frac{\left(\max\limits_{n=1..N}\left\{o_k^{(n)}\right\} + \min\limits_{n=1..N}\left\{o_k^{(n)}\right\}\right)}{2}}{\left(\max\limits_{n=1..N}\left\{o_k^{(n)}\right\} - \min\limits_{n=1..N}\left\{o_k^{(n)}\right\}\right)} \tag{12}$$

where ^ denotes the normalized variable, or parameter.

Experiments were performed also for scaling inputs into range [0 ; 1]. Similar results were achieved for the same learning conditions. The first scaling seems to accelerate slightly the convergence while accuracy and relations among sensitivities remain unchanged. If input and output data scaling (12) has been performed before network training, no additional operations on S_{ki} is required and we have

$$\hat{S}_{ki\,,\,avg} \doteq S_{k\,i,\,avg} \tag{13}$$

Note that scaling can be performed either on entries of S or S_{avg}.

In case when network original inputs and outputs are not scaled to the same level as in (12), additional scaling (14) is necessary to allow for accurate comparison among inputs.

$$\hat{S}_{ki\,,\,avg} \doteq S_{k\,i}\frac{\left(\max\limits_{n=1..N}\left\{x_i^{(n)}\right\} - \min\limits_{n=1..N}\left\{x_i^{(n)}\right\}\right)}{\left(\max\limits_{n=1..N}\left\{o_k^{(n)}\right\} - \min\limits_{n=1..N}\left\{o_k^{(n)}\right\}\right)} \tag{14}$$

The significance measure of i–th input, Φ_i, across the entire set $\mathcal{X}$ is now defined as:

$$\Phi_{i\,,\,avg} \doteq \max_{k=1..K}\left\{\hat{S}_{ki\,,\,avg}\right\}, \quad i = 1,\ ...,\ I-1 \tag{15}$$

Obviously, Φ_{abs} and Φ_{max} can be evaluated similarly to Φ_{avg} defined in (15) if other sensitivity measures are used. Note that searching for entries Φ_i, i=1, 2, ..., I–1, as in (15) corresponds to finding norms of column vectors of the normalized MSA sensitivity matrix $\hat{S}_{avg}$. This can be denoted as

$$\| s_i \|_\infty = \max_{k=1...K} \left| \hat{S}_{ki,\,avg} \right|, \quad i = 1,\ 2,\ \ldots I - 1 \tag{16}$$

In order to distinguish inputs with relative high and low importance and rank them properly, entries of Φ_i have to be sorted in descending order so that:

$$\Phi_{i_{m+1}} \le \Phi_{i_m}, \quad m = 1,\ \ldots,\ I - 2 \tag{17}$$

where $\{i_m\}$ is a sequence of sorted inputs. Note that the sensitivity measures S_{ki} and respective input significance measures, Φ_i, are abstract quantities. Practical heuristic algorithms need now be outlined based on which subsequent input pruning decisions can be made. One of such algorithms is outlined below based on the ratios of two neighboring terms of the sequence Φ_{i_m}.

Let us define the measure of gap as (18)

$$g_m \doteq \frac{\Phi_{i_m}}{\Phi_{i_{m+1}}} \tag{18}$$

and then find the largest gap using the formula (19).

$$g_{MAX} \doteq \max_m \{g_m\} \ \text{ and } \ m_{CUT} \doteq m \text{ such that } g_m = g_{MAX} \tag{19}$$

Determining the largest gap, however, does not imply, that all inputs with coefficients lower than $\Phi_{m\,CUT}$ can be pruned. Whether or not inputs selected for pruning are actually contributing much to the neural network performance an additional criterion is necessary. An example of such criterion is given by equation (20) stating that second largest gap, $g_{MAX\,II}$, has to be much smaller than that given by the formula (19). If condition (20) is valid, then the gap found between m_{CUT} and m_{CUT}+1 is large enough.

$$C g_{MAX} > g_{MAX\,II}, \quad \text{where } g_{MAX\,II} \doteq \max_{i_m \neq i_{m_{CUT}}} \{g_m\} \tag{20}$$

Constant C from (20) is chosen arbitrarily within the reasonable range (e.g. C=0.5. The smaller C,

the stronger is the condition for existence of the acceptable gap.) All inputs with index $\{i_{m+1}..i_{I-1}\}$ can be pruned with the smallest loss of information to the MFNN.

The gap method can be also applied for comparison among sensitivities of inputs to each output separately. For this purpose, a set containing candidates for pruning can be created for every output. Final pruning is performed by removing these inputs which can be found in every set determined previously for each output independently.

Obviously, $\mathbf{S}_{avg}$ can be meaningfully evaluated only for well trained neural networks. Despite this disadvantage, proposed criteria can still save computational effort when initial training is performed on smaller, but still representative subset of data. $\mathbf{S}_{avg}$ can then be evaluated based either on the data set used for initial training or on complete data set. Subsequently, newly developed neural networks with appropriate inputs can be retrained using the full set of training patterns with reduced dimension.

The importance of input i can be also determined statistically. The input saliency method referred earlier [9] uses formula (21) and the averaging of the results over multiple training sessions.

$$\Phi_{i,sal} \doteq \sum_{k=1}^{K} \left| \hat{S}_{ki,sal} \right|, \quad i = 1, ..., I-1 \tag{21}$$

The measure $\Phi_{i,sal}$ is called input saliency. The i-th input is considered to be salient if the average saliency calculated over many training sessions is above the upper confidence boundary as described in more detail in [9]. Such statistical approach, although computationally intensive in comparison with proposed heurisitc solution (19) allows for formulation of the theoretical criterion for unimportant feature removal.

NUMERICAL EXAMPLES

A series of numerical simulations was performed in order to verify the proposed perturbation–based approach and the pruning criteria. In the first experiment a training set for a neural network was produced using four inputs $x_1..x_4$ and two outputs o_1 and o_2. Values of output o_1 were correlated with x_1 and x_2, and of output o_2 with x_2 and x_3. Input vectors $\mathbf{x}$ (4x1) were produced using a random

number generator. The expected values of vector **d** (2x1) for the output vector **o** (2x1) were evaluated for each **x** using a known relationship **d=F**(**x**) where **d** is the desired (target) output vector for supervised training. The training set $\mathcal{X}$ consisted of N=81 patterns. A neural network with 4 inputs, 2 outputs and 6 hidden neurons (I=5, J=7, K=2) has been trained for the mean square error defined as in (22)

$$MSE \doteq \sqrt{\frac{\sum_{n=1}^{N}\sum_{k=1}^{K}\left(d_k^{(n)} - o_k^{(n)}\right)^2}{N}} \tag{22}$$

equal 0.001 per input vector. Matrices of sensitivities were subsequently evaluated and $\mathbf{S}_{avg}$ produced at the end of training over the entire input data set $\mathcal{X}$.

The changes of MSA sensitivity entries during learning are presented in Fig. 3. It can be seen that initial sensitivities are low and apparently random positive numbers. During the training some of the average sensitivities $S_{ki,avg}$ increase, while others converge towards low values. An obvious property can be seen that an untrained neural network in the example has per average smaller sensitivities than after the training. Final values of sensitivities of the first output offer hints for deleting x_3 and x_4, and these for the second output indicate that x_1 and x_4 could be deleted. The only input which then shows up in both sensitivity sets candidates for deletion is x_4. Therefore, the fourth input to the network can be eliminated and its dimension reduced to 3 inputs plus bias (I=4).

The new network with three inputs was trained successfully after deleting x_4 from the training data set with the same accuracy. The learning profiles for full and reduced input sets for the same learning conditions are compared in Fig. 4. Not only the network with three inputs trains within a smaller number of cycles, but each learning cycle is performed faster due to the reduced input layer size.

If an input not recommended for pruning is erroneously deleted, the network is not able to learn the data sets. In our example the MSE value remains at the level of approximately 0.24 as it is shown in Fig. 4. Most entries of the sensitivity matrix remain low as shown in Fig. 5, which is indicative of poor network performance. A network erroneously trimmed is not able to learn accurately because some important relationships have been lost after pruning.

The second experiment was performed using a larger network and random data. MFNN had 20 inputs (I=21), 10 hidden neurons (J=11) and 4 outputs (K=4). There were N=500 patterns in the training set. Several additional data sets of the same size have been generated according to the same rule as the training set for performance evaluation. The network was successfully trained to the MSE of 0.15. However, due to the randomness of the data, MSE for additional sets remained at the level of 0.20. All outputs were strongly correlated with inputs x_1, x_2, x_3, x_4, x_6, x_8, and x_9. Input x_6 during data generation was multiplied by random number, while the influence of x_2 and x_4 on outputs can be seen as scaled down in comparison to other inputs.

The input significance measures calculated using formulas (8)–(10) are shown in Fig. 6. After sorting inputs x_2 and x_4 are ranked as even less important than other inputs which are not correlated at all. This occurred because of their low correlation with outputs, and they can be ignored as well as other inputs which show as uncorrelated for given MSE value as a final condition for training. The sequence of significance measures Φ_{avg}, Φ_{abs}, and Φ_{max}, are the same for all proposed coefficients, however, the size of gaps are different in each case.

Table 1 summarizes numerical results. It lists the sequence $\{i_m\}$ and values g_{im}. Note that m_{cut}=5 and g_{max}=3.013. C was selected arbitrarily as 0.5. Note that value C=0.5 would prevent pruning using ϕ_{max} definition. Also note that the maximum method does not provide a clear clue where to locate the gap for purging due to fuzziness of the training data.

The result of initial training is shown in Fig. 7. It can be determined from this figure which inputs should remain active after pruning. The network performance after pruning is shown in Fig. 8. No additional dimension reduction is advisable because no large gap in input importance is found. The speed of training has increased mostly because of the reduction of the MFNN size (input dimension reduced to 25%). The necessary number of training cycles has also decreased, but not so dramatically as in the first experiment.

An alternative approach to the presented perturbation based method is offered through correlation computation. Input significance coefficients $\Phi_{i,cor}$, can be computed from the definition (24) based

on sensitivity matrix entries given by (23).

$$S_{ki,\,cor} \doteq \frac{\sum_{n=1}^{N}\left(o_k^{(n)} - \overline{o}_k\right)\left(x_i^{(n)} - \overline{x}_i\right)}{\sum_{n=1}^{N}\left(o_k^{(n)} - \overline{o}_k\right)\sum_{n=1}^{N}\left(x_i^{(n)} - \overline{x}_i\right)} \tag{23}$$

$$\Phi_{i,\,cor} \doteq \max_{k=1..K}\left\{S_{ki,\,cor}\right\} \tag{24}$$

Note that this approach requires additional computational effort and it makes use of data only. This is in contrast to the proposed method of calculation of sensitivities and input significance coefficients which requires rather the use of the network model and partially data as well (input vectors only). Fig. 9 illustrates that both methods compare consistently with each other and yield comparable results.

Another experiment was performed using the IRIS data set. That set was first published by Fisher [13] and has been used widely as a testbed for statistical analysis techniques. The sepal length, sepal width, petal length, and petal width were measured on 50 iris specimens from each of 3 species, Iris setosa, Iris versicolor, and Iris virginica. The data set was divided randomly into training data set containing 100 entries, and testing data set containing remaining 50 entries.

As mentioned, the proposed sensitivity method does not apply directly to neural network classifiers, but can still offer guidelines as to the ranked importance of inputs. Classifier in this example was trained for desired output values of –0.5 and 0.5. Although placing neuron outputs outside of the saturation region deteriorates the classifier's performance, it allows to use sensitivity method for input pruning.

Fig. 10 summarizes the result of computational experiment. As seen from Fig. 10a, pruning a single input, #1, causes the increase of error to 10%, while removing inputs #3 and #4 leads to error of 18 and 30%, respectively. In addition, removing input #2 makes it impossible to train the classifier. The pruning algorithm results in significance coefficients as in Fig. 10b, and the gap sizes as in Fig. 10c. After sorting coefficients Φ_i into Φ_{im} and evaluating gap sizes g_m m_{CUT} was set to 3 accord-

ing to the formula (19). Input number i_4=1 can be pruned because condition (20) is satisfied. Correlation analysis performed on IRIS data has not lead to clear indication which inputs can be pruned (see Fig. 11).

CONCLUSIONS

Using the perturbation–based sensitivity approach for input layer pruning seems particularly useful when network training involves large amount of redundant data. In the first phase, network can be pre–trained until the training error has decreased satisfactorily. Then, sensitivity matrices can be evaluated and dimension of the input layer possibly reduced. Training can subsequently be resumed until the training error reduces to an acceptable value. This process can be repeated, however, usually only the first execution yields significant improvement. Numerical experiments indicate that an effort of further network retraining beyond the first pass can be too high in comparison to benefits of further minimization.

Should the redundancy in training data vectors exist, the proposed approach based on the average sensitivity matrices for input data pruning allows for building more efficient perceptron network models. This can be achieved at a relatively low computational cost and based on heuristic pruning criteria outlined in the paper. The approach proposed here is somewhat similar to the principal component analysis in the sense that it detects directions of basis vectors and their relative importance. In contrast to the eigenanalysis for the largest eigenvectors, basis vectors here are fixed. They correspond to the basis vectors in which the original training data are formulated. As such, the approach is aimed at identifying basis vectors yielding minimal projections in the fixed input space dimensions.

The applicability and significance of the presented method is mostly for continuous and differentiable mappings. The method would be even more useful if it allowed additionally merging inputs which are totally correlated on an input set while yielding same target responses. In addition, further extension of the proposed sensitivity–based input pruning approach for binary output networks such as classifiers and binary encoders would be desirable.

ACKNOWLEDGEMENTS: The authors express their thanks to the reviewers for their critical and constructive suggestions which have resulted in significant improvement of the paper.

BIBLIOGRAPHY

[1] K. M. Hornik, M. Stinchcombe, H. White, Multilayer Feedforward Networks Are Universal Approximators, *Neural Networks* 2 (1989) 359–366.

[2] K. I. Funahashi, On the Approximate Realization of Continuous Mappings by Neural Networks, *Neural Networks* 2 (1989) 183–192.

[3] J. M. Zurada, *Introduction to Artificial Neural Systems* (West Publishing Company, St. Paul, Minn., 1992).

[4] P. A. Devijver, J. Kittler, *Pattern Recognition: A Statistical Approach* (Prentice-Hall, Englewood Cliffs, NJ, 1982)

[5] E. D. Karnin, A Simple Procedure for Pruning Back–Propagation Trained Neural Networks, *IEEE Trans. on Neural Networks* 1 (2) (1990).

[6] M. C. Mozer, P. Smolensky, Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment, in: D. S. Touretzky, ed. *Advances in Neural Information Processing I (*Morgan Kaufmann, 1989) 107–115.

[7] J. Y. Choi, C. H. Choi, Sensitivity Analysis of Multilayer Perceptron with Differentiable Activation Functions, *IEEE Transactions on Neural Networks* 3 (1) (1992) 101–107.

[8] D. W. Ruck, S. K. Rogers, M. Kabrisky, Feature Selection using a Multilayer Perceptron, *Neural Network Comp.* 20 (1990) 40–48.

[9] L. M. Belue, K. W. Bauer, Jr., Determining Input Features for Multilayer Perceptrons, *Neurocomputing* 7 (1995) 111–121.

[10] L. Fu, T. Chen, Sensitivity Analysis for Input Vector in Multilayer Feedforward Neural Networks, *in: Proc. of IEEE International Conference on Neural Networks* 1 (San Francisco, CA, March 28–April 1, 1993) 215–218.

[11] J. M. Zurada, A. Malinowski, I. Cloete, Sensitivity Analysis for Pruning of Training Data in Feedforward Neural Networks, in: *Proc. of First Australian and New Zealand Conference on Intelligent Information Systems* (Perth, Western Australia, December 1–3, 1993) 288–292.

[12] J. M. Zurada, A. Malinowski, I. Cloete, Sensitivity Analysis for Minimization of Input Data Dimension for Feedforward Neural Network, in: *Proc. of IEEE International Symposium on Circuits and Systems* (London, May 28–June 2, 1994) 447–450.

[13] Fisher R. A. The Use of Multiple Measurements in Taxonomic, *Annals of Eugenics* (7) (1936) 179–188.

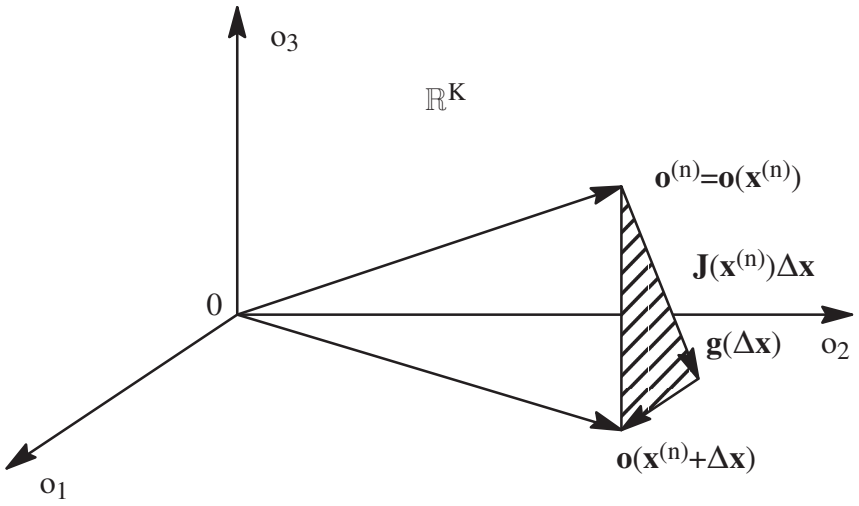

Fig. 1. Geometrical interpretation of output disturbance due to the input disturbance $\Delta\mathbf{x}$

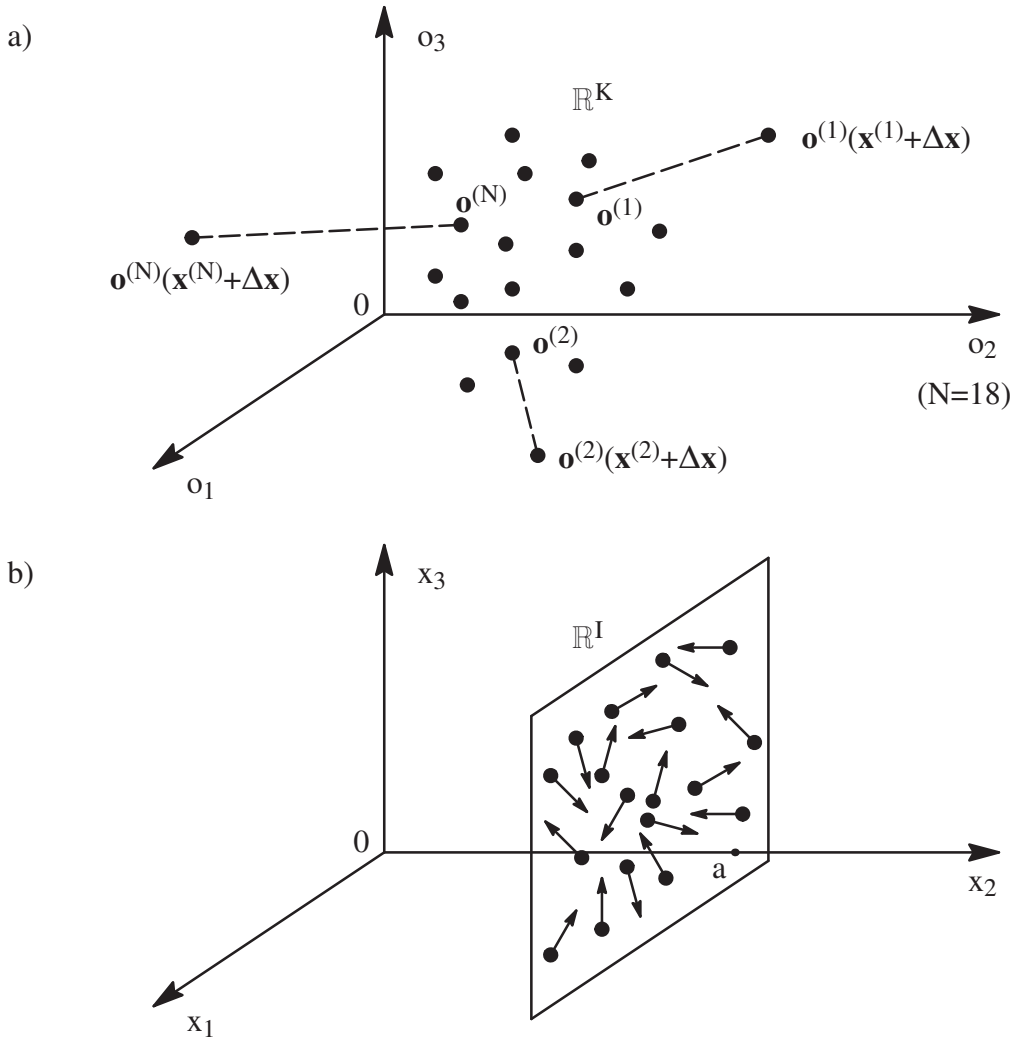

Fig. 2. Perturbation impact in a) output space
b) input space when all output changes are constant in x_2.

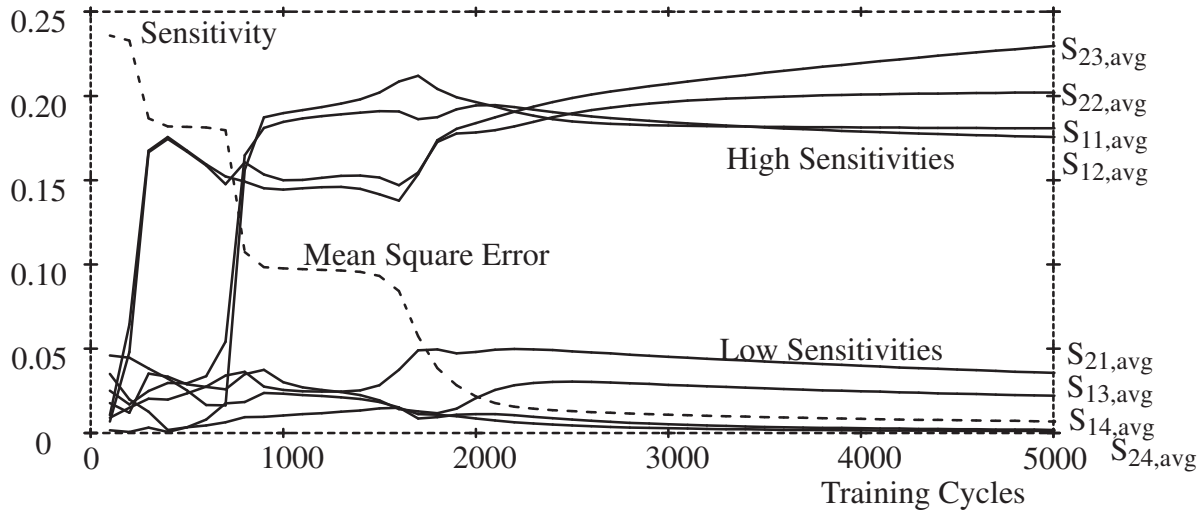

Fig. 3. Sensitivity profile during training for the full training set.

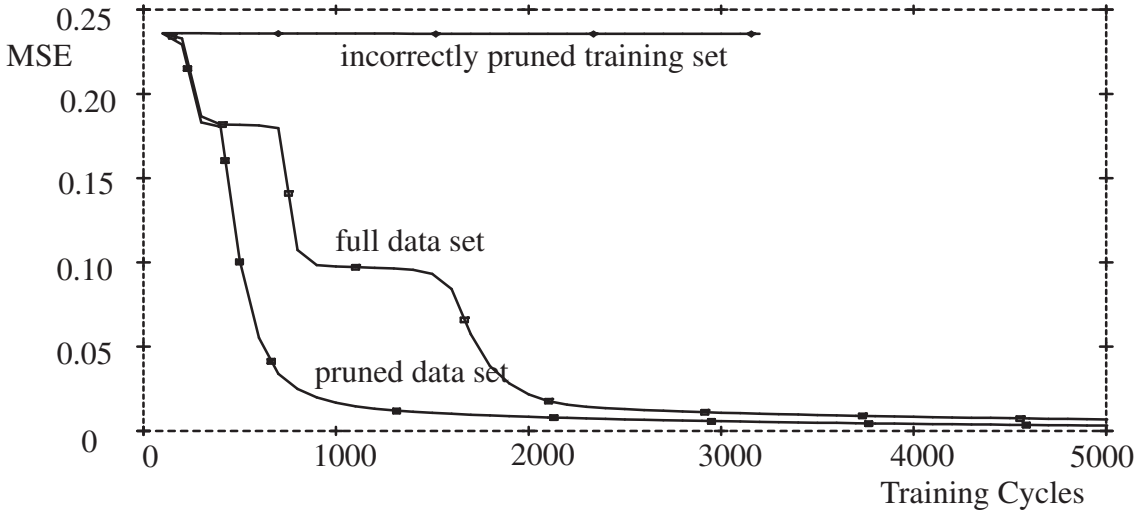

Fig. 4. Learning profile for full and pruned training sets.

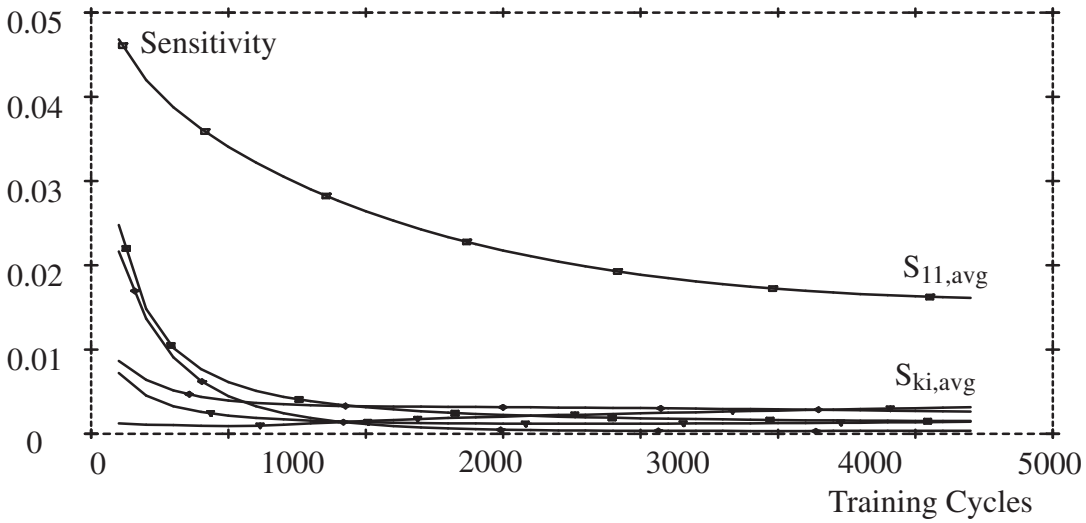

Fig. 5. Sensitivity profile during training for incorrectly trimmed training set.

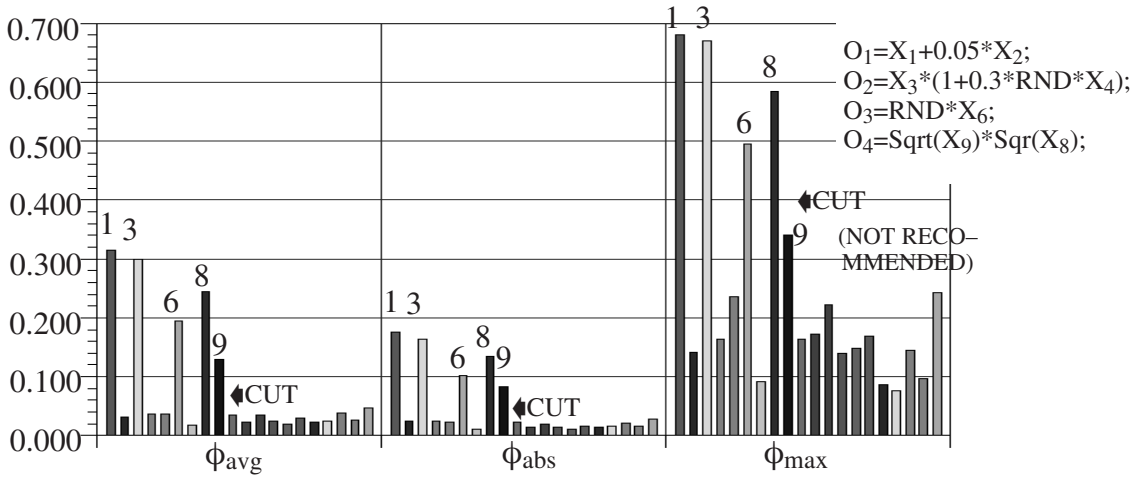

Fig. 6. Input significance ϕ evaluated using different overall sensitivities (8)–(10) and pruning criterion (20). (RND is a random function in [–1; 1].)

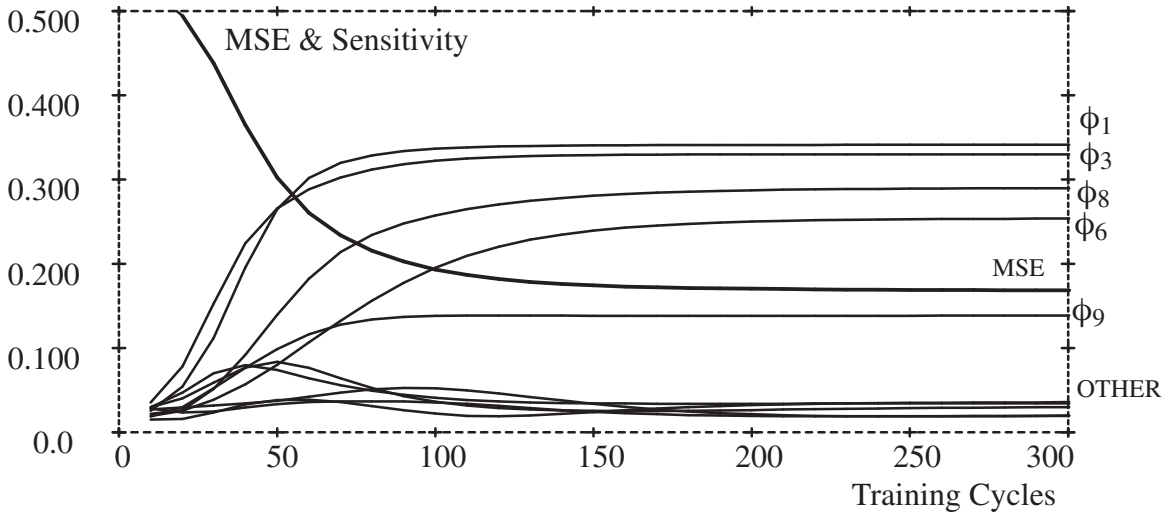

Fig. 7. Input significance ϕ_{avg} changes during training for the full training set.

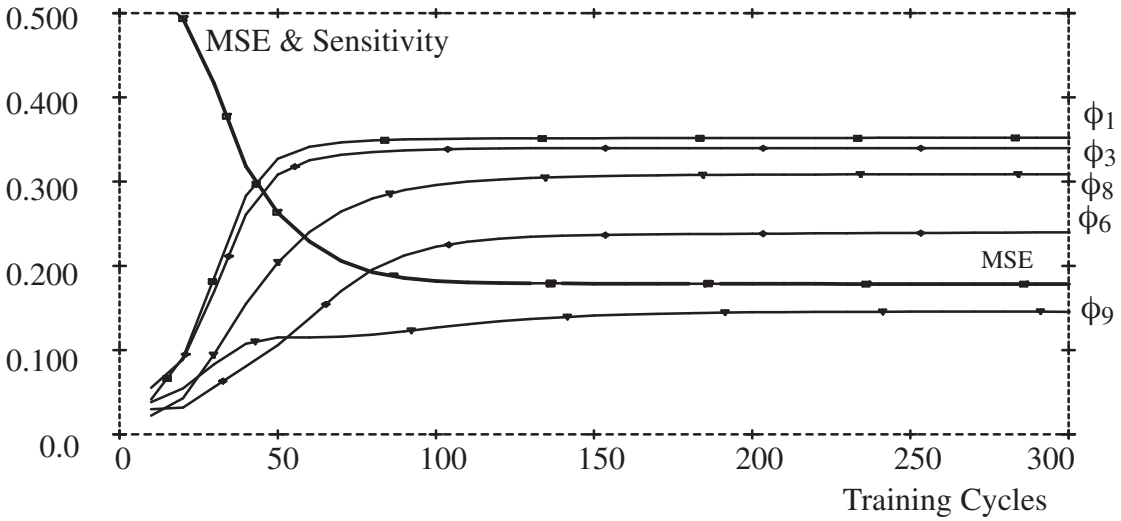

Fig. 8. Input significance ϕ_{avg} changes during training for the pruned training set.

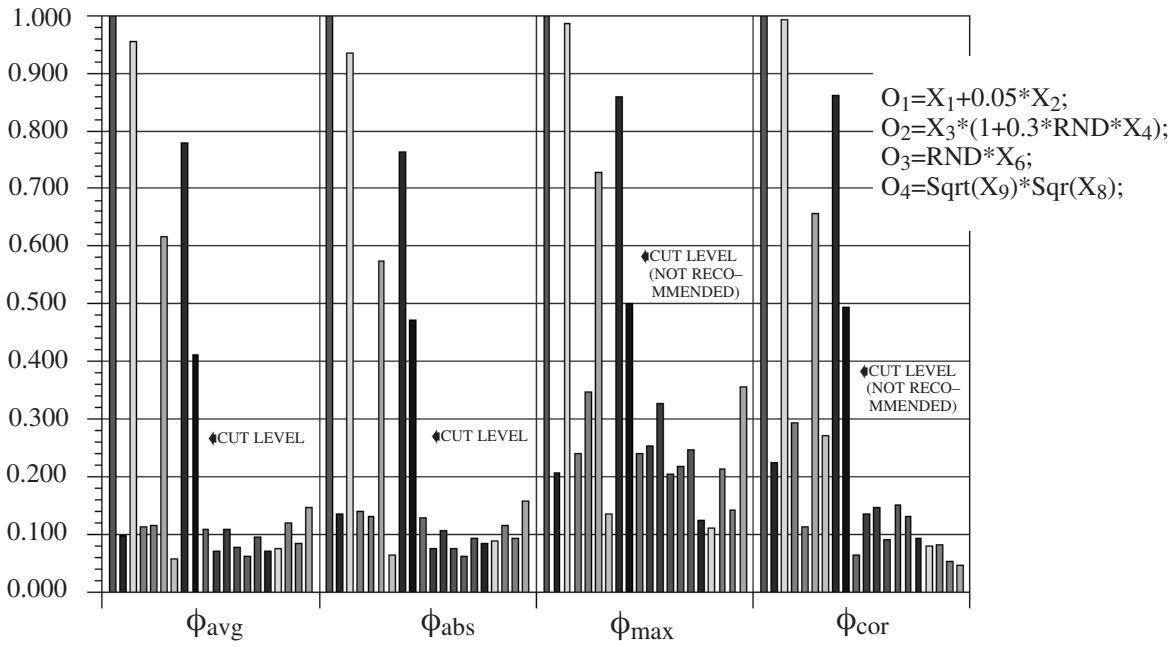

Fig. 9. Normalized input significance coefficients ϕ for different sensitivities (8)–(10) and pruning criterion (20) in comparison to significance coefficients evaluated using standard correlation method ϕ_{cor}.
Significance coefficients are normalized.

(RND is a random function with uniform distribution in [–1; 1].)

$\{i_m\}$	$\phi_{avg,im}$	g_{im}	$\{i_m\}$	$\phi_{abs,im}$	g_{im}	$\{i_m\}$	$\phi_{mx,im}$	g_{im}	$\{i_m\}$	$\phi_{cor,im}$	g_{im}
1	0.314	1.047	1	0.175	1.071	1	0.679	1.015	1	0.998	1.006
3	0.299	1.227	3	0.164	1.223	3	0.669	1.146	3	0.992	1.153
8	0.244	1.371	8	0.134	*1.329*	8	0.584	1.181	8	0.861	1.310
6	0.194	*1.497*	6	0.101	1.218	6	0.494	***1.454***	6	0.657	1.333
9	0.129	***3.013***	9	0.082	***2.979***	9	0.340	1.402	9	0.493	***1.681***
20	0.043	1.132	20	0.027	1.118	20	0.242	1.030	4	0.293	1.081
18	0.038	1.034	4	0.024	1.040	5	0.235	1.061	7	0.271	1.207
5	0.036	1.023	2	0.023	1.029	12	0.222	1.284	2	0.224	*1.476*
4	0.036	1.037	5	0.023	1.029	11	0.172	1.026	14	0.152	1.027
10	0.034	1.001	10	0.022	1.100	15	0.168	1.030	12	0.148	1.088
12	0.034	1.110	18	0.020	1.100	4	0.163	1.000	11	0.136	1.038
2	0.031	1.022	12	0.018	1.130	10	0.163	1.105	15	0.131	1.147
15	0.030	1.134	15	0.016	1.007	14	0.148	1.018	5	0.114	1.225
19	0.026	1.106	19	0.016	1.052	18	0.145	1.029	16	0.093	1.024
13	0.024	1.025	17	0.015	1.034	2	0.141	1.013	13	0.091	1.108
17	0.023	1.054	16	0.015	1.121	13	0.139	*1.447*	18	0.082	1.027
16	0.022	1.005	11	0.013	1.003	19	0.096	1.046	17	0.080	1.236
11	0.022	1.121	13	0.013	1.167	7	0.092	1.078	10	0.064	1.186
14	0.020	1.108	7	0.011	1.034	16	0.085	1,137	19	0.054	1.147
7	0.018	–	14	0.011	–	17	0.075	–	20	0.047	–

m_{cut}=5	m_{cut}=5	m_{cut}=none	m_{cut}=none
g_{MAX}=3.013 at m=5	g_{MAX}=2.979 at m=5	g_{MAX}=1.454 at m=4	g_{MAX}=1.681 at m=5
g_{MAXII}=1.497 at m=4	g_{MAXII}=1.329 at m=4	g_{MAXII}=1.447 at m=16	g_{MAXII}=1.476 at m=8
g_{MAXII}/g_{MAX}=0.497<C	g_{MAXII}/g_{MAX}=0.446<C	g_{MAXII}/g_{MAX}=0.995>C	g_{MAXII}/g_{MAX}=0.878>C

Tab. 1. Intermediate results of the pruning algorithm (Ref. to Figs. 6, 7, and 9).

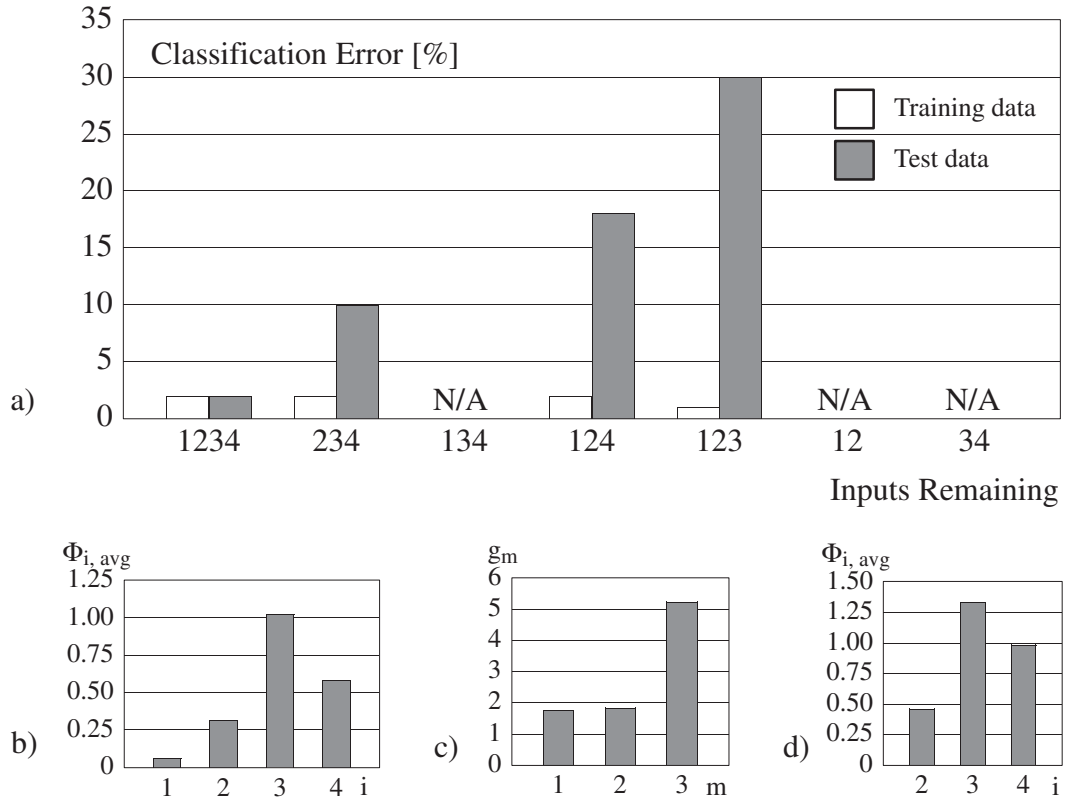

Fig. 10 Neural network training using IRIS data set and termination conditions set by MSE
a) percentage of misclassification for training and testing data sets, MSE=0.05
b) coefficients Φ_i for the complete training data set, MSE=0.01
c) gap sizes for sorted input significance coefficients (b), MSE=0.01
d) coefficients Φ_i after removing input #1, MSE=0.05

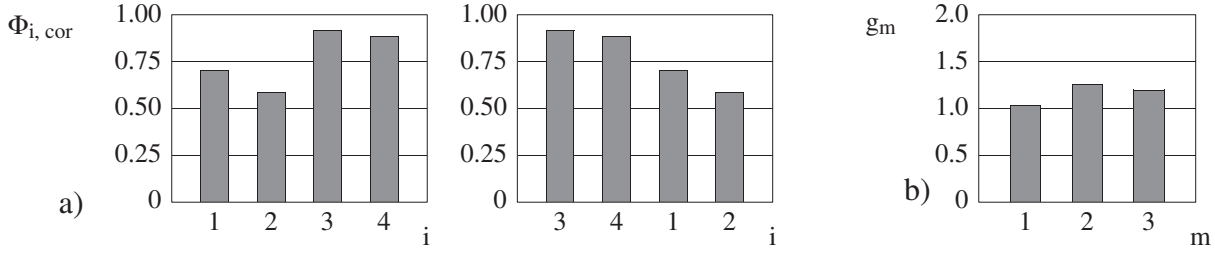

Fig. 11 Correlation between inputs and outputs for IRIS training data set
a) input significance coefficients for the complete training data set
b) gap values for sorted inputs (a).