A Homotopy Method for Training Neural Networks

Markus Lendl, Rolf Unbehauen and Fa-Long Luo

Lehrstuhl für Allgemeine und Theoretische Elektrotechnik, Universität Erlangen-Nürnberg, Cauerstraße 7, D–91058 Erlangen, Germany

Abstract

In many fields of signal processing feed-forward neural networks, especially multilayer perceptron neural networks (MLP-NNs), are used as approximators. On the grounds of expedience the parameter (weight) adaptation process (training) is formulated as an optimization procedure solving a conventional nonlinear regression problem, which is a very important practical task. Thus the presented theory can easily be adapted to any similar problem.

This paper presents an innovative approach to minimize the training error. After a theoretic foundation we will demonstrate that employing the homotopy method (HM) in a second order optimization technique leads to much better convergence properties than direct methods. Simulation results on various examples illustrate excellent robustness concerning the initial values of the weights and less overall computational costs.

Even though this paper addresses learning of neural networks it is outlined in general manner as far as possible motivating to apply the homotopy method to any related parameter optimization problem in signal processing.

Zusammenfassung

In vielen Bereichen der Signalverarbeitung werden neuronale Netzwerke, insbesondere Mehrschicht-Perzeptren, als Approximatoren eingesetzt. Die dazu notwendige Anpassung der Parameter (Gewichte), die man auch Training nennt, wird zweckmäßigerweise als übliches nichtlineares Regressionsproblem formuliert. Da derartige Probleme eine große praktische Bedeutung haben, kann die hier aufgezeigte Theorie häufig auf ähnliche Aufgabenstellungen angewandt werden.

Dieser Aufsatz beschreibt einen neuen Ansatz, beim Training die Fehlerfunktion zu minimieren. Nach einer theoretischen Vorbereitung wird gezeigt, daß die Verwendung einer Homotopie-Strategie zusammen mit einem Optimierungsverfahren zweiter Ordnung wesentlich bessere Konvergenzeigenschaften aufweist als das Optimierungsverfahren alleine. Simulationsergebnisse untermauern die große Unempfindlichkeit gegenüber Parameterstartwerten und damit letztendlich den geringeren numerischen Aufwand.

Preprint submitted to Elsevier Preprint *29th September 1997*

Obwohl in diesem Aufsatz speziell das Training neuronaler Netzwerke behandelt wird, ist der theoretische Teil so allgemein wie möglich gehalten, um in einfacher Weise Homotopie-Verfahren auf andere verwandte Problemstellungen aus dem Bereich der Signalverarbeitung anwenden zu können.

Résumé

résumé en français

Key words: second order learning, homotopy, Gauss-Newton method, multilayer perceptron

1 Introduction

Multilayer perceptron neural networks (MLP-NNs) are being exploited in an increasing number of applications in signal processing as approximators. The ability of such networks to generate a large class of nonlinear multivariate mappings [10] [13] [5] as well as their speed in forward mode and online adaptability motivate researcher to prefer them to classical approaches.

After having fixed the structure of the neural network, i.e. the number of neurons in the different layers, one challenging problem remains: how to find optimal weights according to the given data? There has been much discussion about the so-called learning or training process and meanwhile a vast literature exists. The different methods can be coarsely classified into first and second order techniques using only gradient or additional curvature information [19] [2]. Nevertheless the problem has not really been solved entirely satisfactorily. Especially slow convergence, reaching an insufficient good local minimum and high sensitivity to the initial values of the weights cause several restarts of the training process and unacceptable high computational costs.

In our paper we first give a short introduction to MLP-NNs and then state the training process as nonlinear regression problem (Section 2). Furthermore we discuss some characteristics to select the most promising optimization technique. We will vote for a new approach that employs a homotopy method with Gauss-Newton corrector steps, which is presented in detail in Section 3. Section 4 shows simulation results that underpin our theoretic statements with illustrating the much better convergence properties of our new homotopy approach in comparison to a direct method. A brief summary is given in Section 5.

2 MLP-NNs and nonlinear least squares

Since this homotopy approach is not limited to a fully connected MLP-NN used for later simulations we only give a short sketch of MLP-NNs for the purpose of notation.

2.1 Structure of a MLP-NN

Let us assume that we want to find an approximation to a function $\boldsymbol{y}(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{R}^{M_0}$ and $\boldsymbol{y} \in \mathbb{R}^{M_L}$ using a MLP-NN [12]. With M_0 and M_L the numbers of neurons in the input and output layer are determined, respectively. The numbers of neurons M_l, $l = 1, \ldots, L-1$, in the hidden layers should be in accordance with the specific approximation problem. In Fig. 1 we choose only one hidden layer ($L = 2$). All layers except the output layer get an additional offset neuron with an unchanged activation of one. Neurons of adjacent layers are connected with weights collected in a weight matrix $\boldsymbol{W}^{[l]}$, $l = 1, \ldots, L$. So we get an output signal $\hat{\boldsymbol{y}} \in \mathbb{R}^{M_L}$ of the network (Fig. 1)

$$\hat{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{\psi}^{[2]}\left(\boldsymbol{W}^{[2]}\boldsymbol{\psi}_e^{[1]}\left(\boldsymbol{W}^{[1]}\boldsymbol{x}_e\right)\right), \tag{2.1}$$

where the weight or parameter vector $\boldsymbol{w}$ includes all the entries of $\boldsymbol{W}^{[l]}$. $\boldsymbol{x}_e$ and $\boldsymbol{\psi}_e$ denote the extended versions of $\boldsymbol{x}$ and $\boldsymbol{\psi}$ defined by

$$\boldsymbol{x}_e := \begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix} \tag{2.2}$$

and

$$\boldsymbol{\psi}_e(\cdot) := \begin{bmatrix} 1 \\ \boldsymbol{\psi}(\cdot) \end{bmatrix}, \tag{2.3}$$

with the sigmoidal activation function

$$\psi_i(u_i) = \frac{1}{1 + \mathrm{e}^{-u_i}}. \tag{2.4}$$

For a detailed discussion of selecting a problem dependent structure of the neural network see [4] and [12].

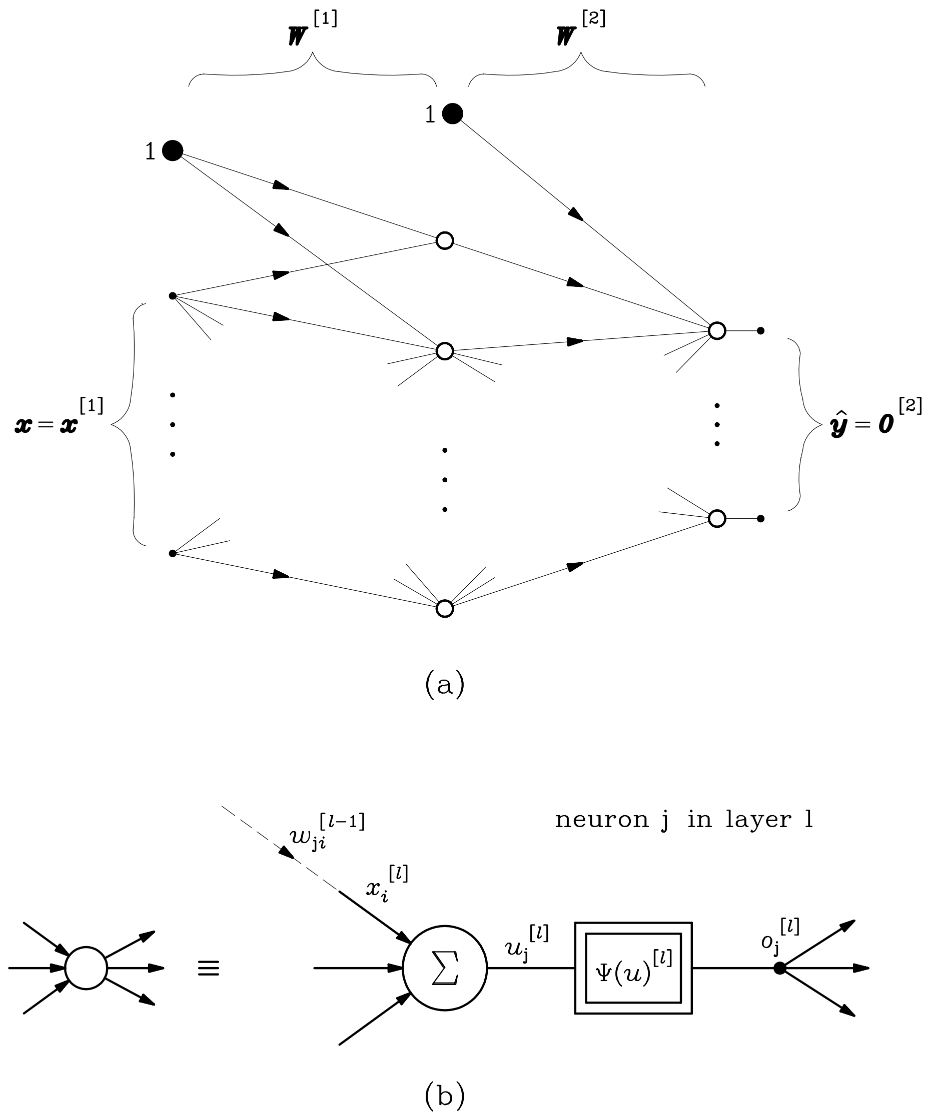


Figure 1. Structure of the MLP-NN (a); a single neuron (b)

2.2 Supervised learning: solving an approximation problem

Now we will state how to adapt the weights to get them "optimal".

Assume that we want to approximate an underlying unknown function $\boldsymbol{y}(\boldsymbol{x})$. All we have is a set $\mathbb{S}$ of P (maybe noisy) samples of $\boldsymbol{y}(\boldsymbol{x})$

$$\mathbb{S} := \left\{ \left(\boldsymbol{x}^{(p)}, \boldsymbol{y}^{(p)} \right) \right\}_{p=1}^{P}, \tag{2.5}$$

with $\boldsymbol{x}^{(p)}$ an input vector and $\boldsymbol{y}^{(p)} = \boldsymbol{y}(\boldsymbol{x}^{(p)})$ the corresponding desired output vector. Let us call $\boldsymbol{w} = \boldsymbol{w}^*$ the optimal parameter vector if $\boldsymbol{w}^*$ minimizes the error or objective function

$$E(\boldsymbol{w}, \mathbb{S}) := \frac{1}{2} \sum_{p=1}^{P} \left\| \boldsymbol{y}^{(p)} - \hat{\boldsymbol{y}}(\boldsymbol{x}^{(p)}, \boldsymbol{w}) \right\|_2^2, \tag{2.6}$$

where $\|\cdot\|_2$ denotes the Euclidean norm. There is much discussion about the choice of an appropriate error function. The sum of squares is very common, easy to use (smooth!) and has a wide application in statistics (maximum likelihood estimation [3] [20] [8]).

For convenience we simplify the notation by defining the residual vector for each sample p

$$\tilde{\boldsymbol{r}}^{(p)}(\boldsymbol{w}) := \boldsymbol{y}^{(p)} - \hat{\boldsymbol{y}}(\boldsymbol{x}^{(p)}, \boldsymbol{w}) \tag{2.7}$$

and the residual vector including all samples

$$\boldsymbol{r}(\boldsymbol{w}, \mathbb{S}) := \left[\tilde{\boldsymbol{r}}^{(1)\,\mathrm{T}} \ldots \tilde{\boldsymbol{r}}^{(P)\,\mathrm{T}} \right]^{\mathrm{T}}. \tag{2.8}$$

Then (2.6) results in

$$E(\boldsymbol{w}, \mathbb{S}) = \frac{1}{2} \sum_{p=1}^{P} \left\| \tilde{\boldsymbol{r}}^{(p)}(\boldsymbol{w}) \right\|_2^2 = \frac{1}{2} \left\| \boldsymbol{r}(\boldsymbol{w}, \mathbb{S}) \right\|_2^2. \tag{2.9}$$

Omitting the argument $\mathbb{S}$ in the following we get the unconstraint optimization problem

$$\text{find} \quad \left\{ \boldsymbol{w}^* \mid E(\boldsymbol{w}^*) = \min_{\boldsymbol{w}} \right\} \tag{2.10}$$

and the whole battery of derivative-free, gradient, conjugate gradient, secant and Newton-type methods can be applied [7] [9].

In order to relate these optimization techniques with the training of neural networks let us mention some literatures for supervised learning. The basic back-propagation algorithm (a gradient decent method) splits into many modifications (for a survey see [19]). But also second order methods are found to be useful [2] [11] [3].

For choosing an appropriate optimization method we take a closer look at some characteristics of the problem (2.10).

For MLP-NNs Owens and Filkin [17] found that the unconstraint optimization problem (2.10) is ill-conditioned for a wide range of $\boldsymbol{w}$. This means that the eigenvalues of the local Hessian matrix

$$\boldsymbol{H}(\boldsymbol{w}) := \frac{\partial^2 E}{\partial \boldsymbol{w}^2}(\boldsymbol{w}) \tag{2.11}$$

differ in orders of magnitude or even $\boldsymbol{H}(\boldsymbol{w})$ is positive semidefinite or indefinite. Therefore gradient based methods show slow progress in reducing the error after a while. To guarantee a high convergence rate, standard literatures in numerical mathematics (e.g. see [7], [9] and [18]) suggest to involve the second order information and employ a Newton-Raphson, Gauss-Newton, secant or at least a conjugate gradient method.

Let us consider the Newton-Raphson formula

$$\boldsymbol{w}[n+1] = \boldsymbol{w}[n] - \boldsymbol{H}^{-1}(\boldsymbol{w}[n])\ \boldsymbol{g}(\boldsymbol{w}[n]), \tag{2.12}$$

with the (locally) positive definite Hessian matrix $\boldsymbol{H}(\boldsymbol{w})$ (cf. (2.11)) and the gradient vector

$$\boldsymbol{g}(\boldsymbol{w}) := \left[\frac{\partial E}{\partial \boldsymbol{w}}(\boldsymbol{w})\right]^{\mathrm{T}}, \tag{2.13}$$

which is quadratically convergent in a neighbourhood of a (local) minimizer $\boldsymbol{w}^*$ of $E(\boldsymbol{w})$. For a least squares problem (2.12) yields to [9]

$$\boldsymbol{w}[n+1] = \boldsymbol{w}[n] - \left[\boldsymbol{J}^{\mathrm{T}}(\boldsymbol{w}[n])\boldsymbol{J}(\boldsymbol{w}[n]) + \boldsymbol{S}(\boldsymbol{w}[n])\right]^{-1}\boldsymbol{g}(\boldsymbol{w}[n]), \tag{2.14}$$

where the matrix

$$\boldsymbol{S}(\boldsymbol{w}) := \sum_{i=1}^{M_L P} r_i(\boldsymbol{w})\,\frac{\partial^2 r_i}{\partial \boldsymbol{w}^2}(\boldsymbol{w}) \tag{2.15}$$

contains the second derivatives and

$$\boldsymbol{J}(\boldsymbol{w}) := \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{w}}(\boldsymbol{w}) \tag{2.16}$$

denotes the Jacobian. From (2.14) and (2.15) we can see that in the case of sufficient small residuals we get a good approximation for the Hessian matrix

$$\boldsymbol{H}(\boldsymbol{w}) \approx \hat{\boldsymbol{H}}(\boldsymbol{w}) = \boldsymbol{J}^{\mathrm{T}}(\boldsymbol{w})\boldsymbol{J}(\boldsymbol{w}) \tag{2.17}$$

by omitting $\boldsymbol{S}(\boldsymbol{w})$. (2.12) results in

$$\boldsymbol{w}[n+1] = \boldsymbol{w}[n] - \hat{\boldsymbol{H}}^{-1}(\boldsymbol{w}[n])\ \boldsymbol{g}(\boldsymbol{w}[n]), \tag{2.18}$$

which is known as the Gauss-Newton formula.

A second observation has to be treated with care. When to use the Newton-Raphson or Gauss-Newton method we have to guarantee a positive definite Hessian matrix $\boldsymbol{H}$ or $\hat{\boldsymbol{H}}$. This can be done by regularization using the Levenberg modification [14]. Thus (2.18) leads to

$$\boldsymbol{w}[n+1] = \boldsymbol{w}[n] - \left[\hat{\boldsymbol{H}}(\boldsymbol{w}[n]) + \lambda[n]\boldsymbol{I}\right]^{-1} \boldsymbol{g}(\boldsymbol{w}[n]), \tag{2.19}$$

where λ denotes the Levenberg parameter and $\boldsymbol{I}$ the identity matrix. It is somewhat tricky to adapt λ. There are several algorithms motivated by the trust region approach [7], but in our experience the simple Marquardt algorithm [15] works even better in the context of training MLP-NNs.

2.4 Further Discussions

Now we want to conclude the selection process for an appropriate optimization (training) method and motivate the basic idea for further significant improvements.

In the previous section we pointed out that training MLP-NNs results in an ill-conditioned nonlinear least squares optimization problem. This urges to apply a second order technique whereby the Levenberg-modified Gauss-Newton method approximates the Hessian matrix by calculating only first order derivatives and guarantees positive definiteness for $\lambda > 0$. The behaviour of Gauss-Newton is expected to be nearly identical to Newton-Raphson in the case of small residuals. Unfortunately, in general the assumption of small residuals is far from practical conditions. Therefore it is straightforward to split the "direct" (in general large residual) minimization problem (2.10) into several small residual ones, which can be solved by the modified Gauss-Newton method very efficiently. This can be done by applying a homotopy technique.

3 A homotopy method

The basic idea behind a homotopy method (HM) is to construct a modified problem that deforms homotopicly from one with a known solution to the original one. During the deformation process, i.e. the change of an additional introduced parameter σ, a path is defined from the known solution to the solution of the original problem. This path is traced by changing σ step by step and by solving a (small residual) subproblem at each step.

In the following sections we first present a standard approach to the HM and then the modifications for our specific least squares problem.

3.1 Construction of a homotopy (basic idea)

Consider the following system of nonlinear equations

$$\boldsymbol{r}(\boldsymbol{w}) = \mathbf{0}; \ \boldsymbol{r} : \mathbb{R}^q \to \mathbb{R}^q, \tag{3.1}$$

with the set of unknown solutions

$$\{\boldsymbol{w}^* \mid \boldsymbol{r}(\boldsymbol{w}^*) = \mathbf{0}\}. \tag{3.2}$$

Now our goal is to find a homotopy $\mathring{\boldsymbol{r}}(\boldsymbol{w}, \sigma)$, [1] $\mathring{\boldsymbol{r}} : \mathbb{R}^{q+1} \to \mathbb{R}^q$, $\sigma \in [0, 1]$, for $\boldsymbol{r}(\boldsymbol{w})$ that is subject to the three restrictions:

$$\mathring{\boldsymbol{r}}(\boldsymbol{w}, 0) \equiv \boldsymbol{r}(\boldsymbol{w}), \tag{3.3}$$

$$\mathring{\boldsymbol{r}}(\boldsymbol{w}, 1) = \mathbf{0} \text{ has a known solution } \boldsymbol{w}(1), \tag{3.4}$$

$$\mathring{\boldsymbol{r}}(\boldsymbol{w}, \sigma) \text{ is continuous in } \sigma. \tag{3.5}$$

One way to construct $\mathring{\boldsymbol{r}}(\boldsymbol{w}, \sigma)$ is to choose a *convex* homotopy [1]

$$\mathring{\boldsymbol{r}}(\boldsymbol{w}, \sigma) = (1 - \sigma)\boldsymbol{r}(\boldsymbol{w}) + \sigma \boldsymbol{s}_r(\boldsymbol{w}), \tag{3.6}$$

where $\boldsymbol{s}_r : \mathbb{R}^q \to \mathbb{R}^q$ is to choose so that $\boldsymbol{s}_r(\boldsymbol{w}) = \mathbf{0}$ has known or easy to get solutions. With $\boldsymbol{s}_r(\boldsymbol{w}) := \boldsymbol{r}(\boldsymbol{w}) - \boldsymbol{r}(\boldsymbol{w}_0)$, $\boldsymbol{w}_0$ a fixed initial value, we receive the *global* homotopy [1]

$$\mathring{\boldsymbol{r}}(\boldsymbol{w}, \sigma) = \boldsymbol{r}(\boldsymbol{w}) - \sigma \boldsymbol{r}(\boldsymbol{w}_0). \tag{3.7}$$

[1] Some notational conventions: A solution $\boldsymbol{w}^*$ of $\mathring{\boldsymbol{r}}(\boldsymbol{w}, \sigma) = \mathbf{0}$ is a function of σ. So $\mathring{\boldsymbol{r}}(\boldsymbol{w}, \sigma)$ is short for $\mathring{\boldsymbol{r}}(\boldsymbol{w}(\sigma), \sigma)$.

Once we have found a homotopy $\mathring{\boldsymbol{r}}(\boldsymbol{w},\sigma)$ (probably specific to the problem) we try to trace the implicit defined curve $\boldsymbol{w}^*(\sigma)$ by employing a predictor-corrector continuation method that starts at $\sigma = 1$ and $\boldsymbol{w}^*(1) = \boldsymbol{w}_0$, and ends at $\sigma = 0$ and $\boldsymbol{w}^*(0)$, which is a solution $\boldsymbol{w}^*$ of our original problem (3.2). This implies to choose a sequence $\left\{\sigma[k]\right\}_{k=0}^{K}$, with $\sigma[0] = 1$, $\sigma[K] = 0$ and $\sigma[k] \leq \sigma[k-1]$. A simple approach is to leave $\Delta\sigma := \sigma[k] - \sigma[k-1]$, $k = 1, \ldots, K$, fixed. This yields to

$$\sigma[k] = \frac{k}{K}\,. \tag{3.8}$$

Every $\sigma[k]$ generates a new subproblem k

$$\mathring{\boldsymbol{r}}\left(\boldsymbol{w}(\sigma[k]), \sigma[k]\right) = \mathbf{0}. \tag{3.9}$$

For the first one with $\sigma[0] = 1$ we know the solution $\boldsymbol{w}^*(\sigma[0]) := \boldsymbol{w}^*(1) = \boldsymbol{w}_0$. This solution is now used as an initial value for the next subproblem $\mathring{\boldsymbol{r}}(\boldsymbol{w},\sigma[1]) = \mathbf{0}$ (prediction step). Employing an (iterative) equation solver, e.g. a Newton technique, we get back sufficiently close to the curve $\boldsymbol{w}^*(\sigma)$ after N_k (corrector) steps. The optimal solution $\boldsymbol{w}^*(\sigma[1]) = \boldsymbol{w}[1, N_k]$ [2] defines the starting point $\boldsymbol{w}[2,0]$ of the subproblem corresponding to $\sigma[2]$ and so on. Fig. 2 illustrates the proceeding of this predictor-corrector algorithm for a one-dimensional parameter space ($\boldsymbol{w} \in \mathbb{R}^1$) and an unique curve $\boldsymbol{w}^*(\sigma)$.

It should be noted that individual termination criteria can apply for different continuation steps. [3] Especially the last step, where we solve the original problem (3.1), may urge a special selection in order to reach a sufficient good $\boldsymbol{w}^*$.

3.2 A homotopy for least squares problems

When we apply a Newton-type method to an optimization problem such as (2.10) we seek $\{\boldsymbol{w}^* | \, \boldsymbol{g}(\boldsymbol{w}^*) = \mathbf{0}\}$, with the gradient vector $\boldsymbol{g}$ defined in (2.13). This means that if we replace $\boldsymbol{r}(\boldsymbol{w})$ by $\boldsymbol{g}(\boldsymbol{w})$ in (3.6) and (3.7) we receive two possible homotopies for our least squares problem:

$$\mathring{\boldsymbol{g}}_1(\boldsymbol{w},\sigma) = (1-\sigma)\boldsymbol{g}(\boldsymbol{w}) + \sigma \boldsymbol{s}_g(\boldsymbol{w}), \tag{3.10}$$

[2] Here $\boldsymbol{w}[k,n]$ denotes the parameter vector $\boldsymbol{w}$ after the corrector step n for the k-th subproblem using $\sigma[k]$. If the specification of the subproblem is not important k is omitted.

[3] The continuation step k subsumes one predictor step and N_k corrector steps.

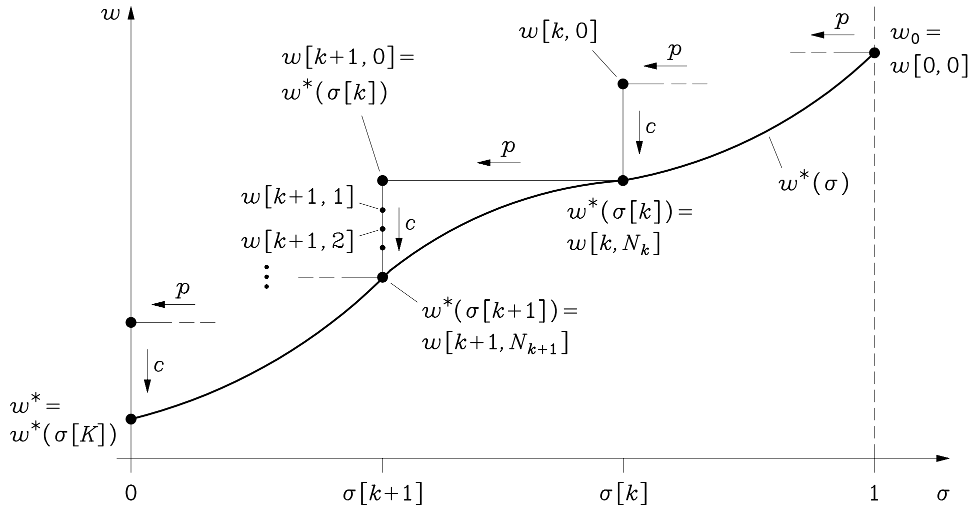


Figure 2. Prediction ("p") and correction ("c") steps for a one-dimensional problem ($\boldsymbol{w} \in \mathbb{R}^1$)

with an appropriate $\boldsymbol{s}_g : \mathbb{R}^q \to \mathbb{R}^q$, and

$$\mathring{\boldsymbol{g}}_2(\boldsymbol{w}, \sigma) = \boldsymbol{g}(\boldsymbol{w}) - \sigma \boldsymbol{g}(\boldsymbol{w}_0). \tag{3.11}$$

A slightly different approach leads to similar expressions. Considering (2.8) and replacing $\boldsymbol{r}(\boldsymbol{w}, \mathbb{S})$ in (2.9) with $\mathring{\boldsymbol{r}}(\boldsymbol{w}, \sigma)$ [4] defined in (3.6) and (3.7), we get a modified minimization problem

$$\text{find} \quad \left\{ \boldsymbol{w}^* \mid \mathring{E}(\boldsymbol{w}, \sigma) := \frac{1}{2} \left\| \mathring{\boldsymbol{r}}(\boldsymbol{w}, \sigma) \right\|_2^2 = \min_{\boldsymbol{w}} \right\}, \tag{3.12}$$

that depends on the homotopy parameter σ. Now we proceed as in Section 3.1. For the first homotopy step set $\sigma[0] = 1$, (3.12) results in a given $\boldsymbol{w}_0$. Then we calculate $\sigma[1]$, e.g. employing (3.8), solve (3.12) with an (iterative) optimization method using $\boldsymbol{w}_0$ as an initial value and get $\boldsymbol{w}^*(\sigma[1])$, and so on until we reach $\boldsymbol{w}^* \equiv \boldsymbol{w}^*(0)$. Fig. 3 compares the structure of the direct and the homotopy driven optimization process.

If we choose a convex homotopy (3.6) the error function

$$\mathring{E}_3(\boldsymbol{w}, \sigma) := \frac{1}{2} \left\| (1 - \sigma) \boldsymbol{r}(\boldsymbol{w}) + \sigma \boldsymbol{s}_r(\boldsymbol{w}) \right\|_2^2 \tag{3.13}$$

[4] $\mathbb{S}$ is omitted for better readability

is to be minimized. Applying a Newton-Raphson method yields to [5] (cf. (2.12))

$$\boldsymbol{w}[n+1] = \boldsymbol{w}[n] - \mathring{\boldsymbol{H}}_3^{-1}(\boldsymbol{w}[n],\sigma)\,\mathring{\boldsymbol{g}}_3(\boldsymbol{w}[n],\sigma), \tag{3.14}$$

with the gradient vector

$$\mathring{\boldsymbol{g}}_3(\boldsymbol{w}[n],\sigma) = \mathring{\boldsymbol{J}}^{\mathrm{T}}(\boldsymbol{w}[n],\sigma)\mathring{\boldsymbol{r}}(\boldsymbol{w}[n],\sigma) \tag{3.15}$$

and the Hessian matrix

$$\begin{aligned}&\mathring{\boldsymbol{H}}_3(\boldsymbol{w}[n],\sigma) = \\ &\quad\Big[\mathring{\boldsymbol{J}}^{\mathrm{T}}(\boldsymbol{w}[n],\sigma)\mathring{\boldsymbol{J}}(\boldsymbol{w}[n],\sigma) + (1-\sigma)\mathring{\boldsymbol{S}}_r(\boldsymbol{w}[n],\sigma) + \sigma\mathring{\boldsymbol{S}}_s(\boldsymbol{w}[n],\sigma)\Big],\end{aligned} \tag{3.16}$$

with

$$\mathring{\boldsymbol{J}}(\boldsymbol{w}[n],\sigma) := \frac{\partial\mathring{\boldsymbol{r}}}{\partial\boldsymbol{w}}(\boldsymbol{w}[n],\sigma) = (1-\sigma)\boldsymbol{J}_r(\boldsymbol{w}[n]) + \sigma\boldsymbol{J}_s(\boldsymbol{w}[n]), \tag{3.17}$$

$$\mathring{\boldsymbol{S}}_r(\boldsymbol{w}[n],\sigma) := \sum_{i=1}^{M_L P} \mathring{\boldsymbol{r}}_i(\boldsymbol{w}[n],\sigma)\,\frac{\partial^2 r_i}{\partial\boldsymbol{w}^2}(\boldsymbol{w}[n]) \tag{3.18}$$

and

$$\mathring{\boldsymbol{S}}_s(\boldsymbol{w}[n],\sigma) := \sum_{i=1}^{M_L P} \mathring{\boldsymbol{r}}_i(\boldsymbol{w}[n],\sigma)\,\frac{\partial^2 s_{r_i}}{\partial\boldsymbol{w}^2}(\boldsymbol{w}[n]). \tag{3.19}$$

The r- and s-Jacobians are defined by

$$\boldsymbol{J}_r(\boldsymbol{w}[n]) := \frac{\partial\boldsymbol{r}}{\partial\boldsymbol{w}}(\boldsymbol{w}[n]) \tag{3.20}$$

and

$$\boldsymbol{J}_s(\boldsymbol{w}[n]) := \frac{\partial\boldsymbol{s}_r}{\partial\boldsymbol{w}}(\boldsymbol{w}[n]). \tag{3.21}$$

For a global homotopy (3.7) we get the error function

$$\mathring{E}_4(\boldsymbol{w}[n],\sigma) := \frac{1}{2}\Big\|\boldsymbol{r}(\boldsymbol{w}[n]) - \sigma\boldsymbol{r}(\boldsymbol{w}_0)\Big\|_2^2, \tag{3.22}$$

[5] see footnote 2

the gradient vector

$$\mathring{\boldsymbol{g}}_4(\boldsymbol{w}[n],\sigma) = \boldsymbol{J}_r^{\mathrm{T}}(\boldsymbol{w}[n])\mathring{\boldsymbol{r}}(\boldsymbol{w}[n],\sigma) \tag{3.23}$$

and the Hessian matrix

$$\mathring{\boldsymbol{H}}_4(\boldsymbol{w}[n],\sigma) = \boldsymbol{J}_r^{\mathrm{T}}(\boldsymbol{w}[n])\boldsymbol{J}_r(\boldsymbol{w}[n]) + \mathring{\boldsymbol{S}}_r(\boldsymbol{w}[n],\sigma), \tag{3.24}$$

with $\boldsymbol{J}_r$ and $\mathring{\boldsymbol{S}}_r$ as in (3.20) and (3.18) respectively. We receive the Newton-Raphson formula if we replace $\mathring{\boldsymbol{g}}_3$ and $\mathring{\boldsymbol{H}}_3$ in (3.14) with $\mathring{\boldsymbol{g}}_4$ and $\mathring{\boldsymbol{H}}_4$ respectively.

In the case of applying the Gauss-Newton method the convex homotopy approach (3.13) finally yields to

$$\begin{aligned}\boldsymbol{w}[n+1] &= \\ &\boldsymbol{w}[n] - \left[\mathring{\boldsymbol{J}}^{\mathrm{T}}(\boldsymbol{w}[n],\sigma)\mathring{\boldsymbol{J}}(\boldsymbol{w}[n],\sigma)\right]^{-1}\mathring{\boldsymbol{J}}^{\mathrm{T}}(\boldsymbol{w}[n],\sigma)\mathring{\boldsymbol{r}}(\boldsymbol{w}[n],\sigma)\end{aligned} \tag{3.25}$$

and for a constant $\boldsymbol{s}_r(\boldsymbol{w}) = \boldsymbol{s}$ in (3.13) we get the same result as for the global homotopy approach (3.22)

$$\boldsymbol{w}[n+1] = \boldsymbol{w}[n] - \left[\boldsymbol{J}_r^{\mathrm{T}}(\boldsymbol{w}[n])\boldsymbol{J}_r(\boldsymbol{w}[n])\right]^{-1}\boldsymbol{J}_r^{\mathrm{T}}(\boldsymbol{w}[n])\mathring{\boldsymbol{r}}(\boldsymbol{w}[n],\sigma). \tag{3.26}$$

It is worthy noting for implementational aspects that a dependency of σ occurs only in the most right hand expression of (3.26) $\mathring{\boldsymbol{r}}(\boldsymbol{w}[n],\sigma)$.

3.3 The tangent type predictor step

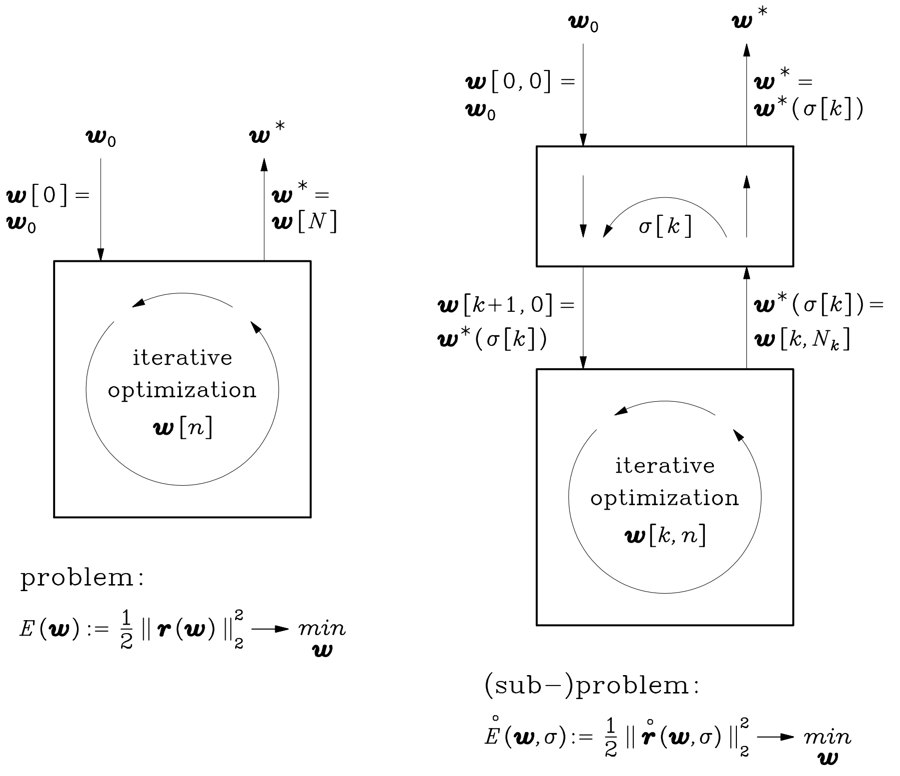
When considering the global homotopy as in (3.11)

$$\mathring{\boldsymbol{g}}(\boldsymbol{w},\sigma) := \boldsymbol{g}(\boldsymbol{w}) - \sigma\,\boldsymbol{g}_0, \quad \mathring{\boldsymbol{g}} : \mathbb{R}^{M_L P+1} \to \mathbb{R}^{M_L P}, \quad \boldsymbol{g}_0 := \boldsymbol{g}(\boldsymbol{w}_0) \tag{3.27}$$

and recalling the predictor step

$$\boldsymbol{w}[k+1,0] = \boldsymbol{w}^*[k] \tag{3.28}$$

as described above, one may argue that a better initial value $\tilde{\boldsymbol{w}}[k+1,0]$ for the following corrector iterations can be found by moving along the tangent direction $\boldsymbol{w}' := \dfrac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\sigma}$ (tangent continuation method) to a prespecified $\sigma[k+1]$ (see Fig. 4), because $\tilde{\boldsymbol{w}}[k+1,0]$ is closer to the curve $\boldsymbol{w}^*(\sigma)$ then $\boldsymbol{w}[k+1,0]$. We will now show that under particular circumstances, which are relevant in



(a) (b)

Figure 3. Direct (a) and homotopy driven (b) optimization

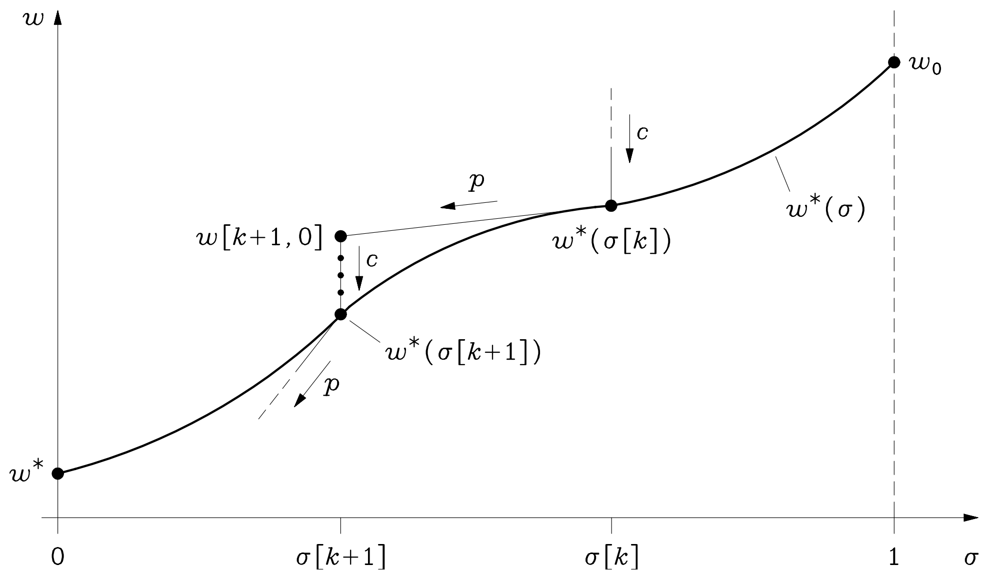


Figure 4. Tangent predictor steps

praxis, a tangent predictor step is adequate to an additional corrector step. We choose

$$\tilde{\boldsymbol{w}}[k+1,0] = \boldsymbol{w}^*(\sigma[k]) + s[k+1]\,\boldsymbol{w}'(\boldsymbol{w}^*(\sigma[k])), \tag{3.29}$$

with the stepsize $s[k+1]$, and receive $\boldsymbol{w}'(\boldsymbol{w})$ by differentiating $\mathring{\boldsymbol{g}}(\boldsymbol{w},\sigma) = \mathbf{0}$ for σ

$$\frac{\mathrm{d}\mathring{\boldsymbol{g}}}{\mathrm{d}\sigma}(\boldsymbol{w},\sigma) = \frac{\partial \mathring{\boldsymbol{g}}}{\partial \boldsymbol{w}}(\boldsymbol{w},\sigma)\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\sigma}(\boldsymbol{w},\sigma) + \frac{\partial \mathring{\boldsymbol{g}}}{\partial \sigma}(\boldsymbol{w},\sigma) = \mathbf{0}. \tag{3.30}$$

With

$$\frac{\partial \mathring{\boldsymbol{g}}}{\partial \boldsymbol{w}}(\boldsymbol{w},\sigma) = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{w}}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w}) \tag{3.31}$$

and

$$\mathring{\boldsymbol{J}}_\sigma(\boldsymbol{w},\sigma) := \frac{\partial \mathring{\boldsymbol{g}}}{\partial \sigma}(\boldsymbol{w},\sigma) = -\boldsymbol{g}_0, \tag{3.32}$$

we obtain

$$\boldsymbol{w}'(\boldsymbol{w}) = \boldsymbol{H}^{-1}(\boldsymbol{w})\,\boldsymbol{g}_0, \tag{3.33}$$

and with respect to (3.29)

$$\tilde{\boldsymbol{w}}[k+1,0] = \boldsymbol{w}^*(\sigma[k]) + s[k+1]\,\boldsymbol{H}^{-1}(\boldsymbol{w}^*(\sigma[k]))\,\boldsymbol{g}_0. \tag{3.34}$$

If we compare this with the first corrector step when applying the Gauss-Newton formula (2.18)

$$\begin{aligned}&\boldsymbol{w}[k+1,1] = \\ &\quad \boldsymbol{w}^*(\sigma[k]) - \hat{\boldsymbol{H}}^{-1}(\boldsymbol{w}^*(\sigma[k]))\left[\boldsymbol{g}(\boldsymbol{w}^*(\sigma[k])) - \sigma[k+1]\,\boldsymbol{g}_0\right]\end{aligned} \tag{3.35}$$

and taking into account that for an optimal $\boldsymbol{w}^*[k]$

$$\mathring{\boldsymbol{g}}(\boldsymbol{w}^*(\sigma[k]),\sigma[k]) = \mathbf{0} \quad\Longleftrightarrow\quad \boldsymbol{g}(\boldsymbol{w}^*(\sigma[k])) = \sigma[k]\,\boldsymbol{g}_0 \tag{3.36}$$

holds, we realize that (3.34) and (3.35) become identical in the case of

$$\hat{\boldsymbol{H}}^{-1}(\boldsymbol{w}^*(\sigma[k])) = \boldsymbol{H}^{-1}(\boldsymbol{w}^*(\sigma[k])) \tag{3.37}$$

and

$$\sigma[k+1] - \sigma[k] = s[k+1]. \tag{3.38}$$

Equation (3.37) is fulfilled when in (3.34) the Hessian matrix is replaced by its Gauss-Newton approximation (2.17) and $s[k+1]$ can always be chosen according to (3.38). This means that we need not to trace tangent type predictor steps for a global homotopy (cf. (3.27)) when using the Gauss-Newton technique for the corrector iterations.

4 Experimental results

In order to get an idea of the performance of the novel homotopy driven training algorithm, we consider two examples. The first one is taken from [11], where the superiority of a direct Gauss-Newton method over back-propagation with momentum and a conjugate gradient technique is outlined. Therefore we focus on a comparison between the direct and a homotopy driven optimization method. The second example deals with a real-world application, the approximation of the inverse of the family of characteristics of a turbidity sensor.

4.1 Problem 1: sinusoidal function

For the first problem, the approximation of a sinusoidal function

$$f(x) = \frac{1}{2} + \frac{1}{4}\sin(3\pi x),$$

we use a 1:15:1 MLP-NN with sigmoidal nonlinearities in the hidden layer and a linear output layer. The training set consists of 81 input/output pairs scattered in the interval $[-1, 1]$. The initial weights are generated uniformly distributed in $[-2, 2]$ and normalized with the algorithm proposed by Nguyen and Widrow [16]. An initial Levenberg parameter $\lambda_0 = 0,01$ and an increase/decrease factor of 3 for the Marquardt formula and a global homotopy based on the residuum (3.22) are used. In addition we found it useful to follow Villiers and Glasser [6] and restrict the corrector iterations to one step. Fig. 5 shows the progress of the training error subject to the original error function (2.6) for a direct Gauss-Newton method, i.e. one continuation step (Fig. 5: (1)), and a homotopy method with three, five, ten and twenty uniformly spaced (cf. (3.8)) continuation steps (Fig. 5: (1), (3), (5), (10), (20)). We have to note that the computational costs for formulating a new subproblem, i.e. the calculation of $\sigma[k]$, can be neglected and thus the number of corrector steps m is an appropriate measure for the numerical expense. During the first few steps the direct method decreases quickly but further progress typically slows down. In contrast to that the continuation technique minimizes subproblems different from the original. Subsequently this leads to a slow decrease of the error (of the original problem) in the beginning but

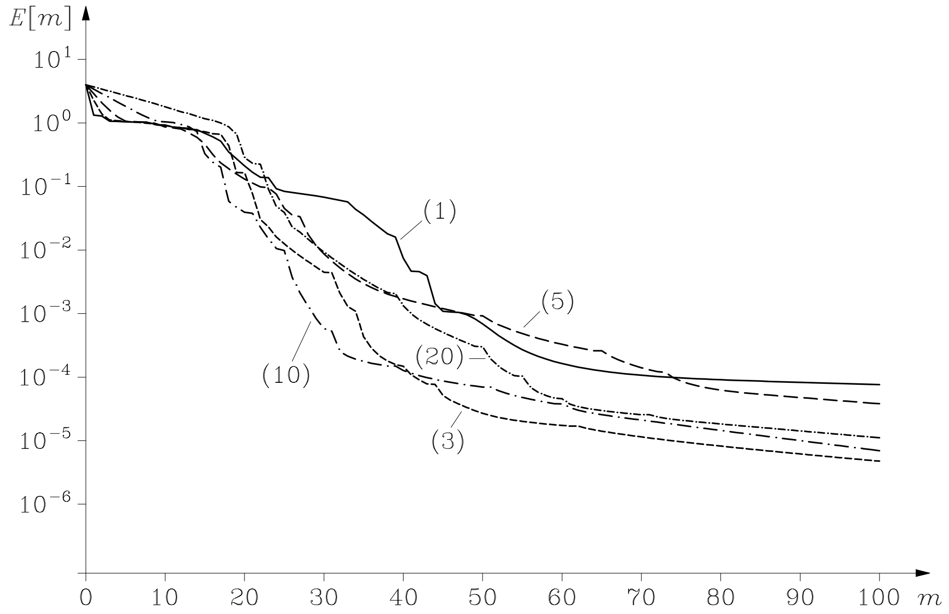


Figure 5. Error E versus corrector steps m for the sin approximation problem

much faster progress when $\sigma = 0$ is reached. Therefore, it is clear that only a "small" number of continuation steps makes sense for this kind of problems.

4.2 Problem 2: turbidity sensor

Consider a sensor that outputs two voltages x_1, x_2 which express the current turbidity t. In a sequence of tests we find a set $\mathbb{S}$ of $P = 51$ training pattern $\{(x_1^{(p)}, x_2^{(p)}, t^{(p)})\}_{p=1}^{P}$. For a faster learning we add a signal

$$x_3^{(p)} = \begin{cases} 0 \ , \ x_1^{(p)} \geq x_2^{(p)} \\ 1 \ , \ x_1^{(p)} < x_2^{(p)} \end{cases} ,$$

that allows to better distinguish small x_1, x_2 at low turbidity from these at high one (Fig. 6). Now we seek an approximation $\hat{t}(\boldsymbol{x}, \mathbb{S})$ to the inverse sensor system. Similar to Problem 1 a 3:10:1 MLP-NN with sigmoidal activation function in the hidden layer and a linear output neuron is used. The Levenberg-modified Gauss-Newton method (as in Problem 1) is applied to a global homotopy based on the gradient (3.11). A typical curve for the true training error (Fig. 7) confirms what we expect from theoretic discussions.

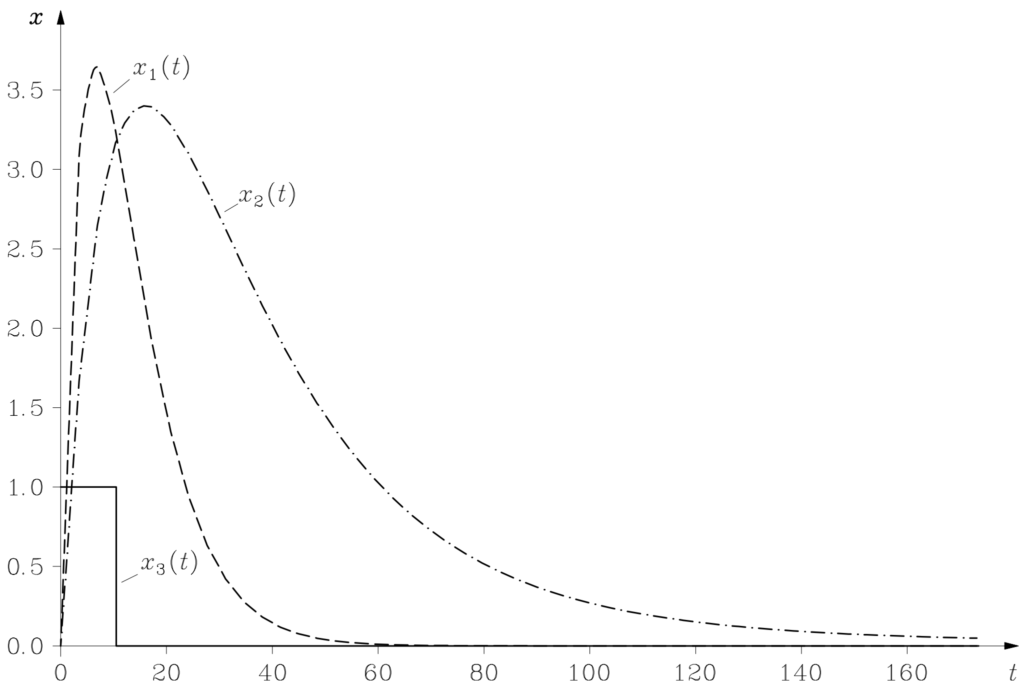


Figure 6. Family of characteristics of a turbidity sensor $\boldsymbol{x}(t)$

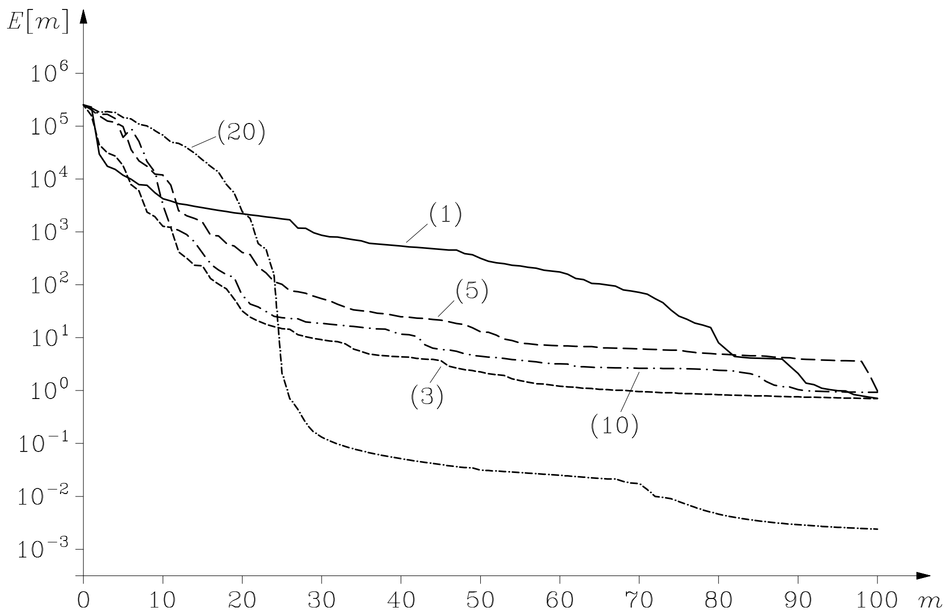


Figure 7. Error E versus corrector steps m for the Problem 2

5 Conclusions

Since MLP-NN are extremely well suited for approximation tasks, they are more and more employed for signal processing tasks. One important problem, the finding of optimal parameters (weights), remains. In this paper the training process is stated as an ill-conditioned least squares optimization problem. Taking these characteristics into account the Levenberg modification of the Gauss-Newton method appears favourable to solve this task. In fact Hagan and Menhaj [11] showed its superiority in comparison with other training algorithms. For the purpose of improving the working conditions of the Gauss-Newton method the generally large residual optimizations problem is split into several small residual subproblems by applying a homotopy method. This results in a further speed up of convergence and thus an additional decrease of overall computational costs.

The focus of this paper is not to discuss the performance of various training algorithms but to show an efficient modification to the standard Gauss-Newton type optimization as special application to neural networks. This is presented in a manner to easily allow an adaptation to any similar optimization problem in signal processing.

6 Acknowledgements

This work was partially supported by the German Research Society (DFG).

References

[1] E. L. Allgower and K. Georg. *Numerical continuation methods.* Springer, 1990.

[2] R. Battiti. First and second order methods for learning: Between steepest descent and Newton's method. *Neural Computation*, 4(2):141–166, 1992.

[3] C. M. Bishop. *Neural networks for pattern recognition.* Oxford Clarendon Press, 1995.

[4] A. Cichocki and R. Unbehauen. *Neural networks for optimization and signal processing.* Teubner, 1993.

[5] G. Cybenko. Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:304–314, 1989.

[6] N. de Villiers and D. Glasser. A continuation method for nonlinear regression. *SIAM J. Num. Anal.*, 18(6):1139–1154, Dez 1981.

[7] J. E. Dennis and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations.* SIAM, 1996.

[8] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis.* Wiley, 1973.

[9] R. Fletcher. *Practical methods of optimization.* Wiley, 1995.

[10] K. Funahashi. On the approximate realization of continuos mappings by neural networks. *Neural Networks*, 2:183–192, 1989.

[11] M. T. Hagan and M. B. Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993, november 1994.

[12] S. Haykin. *Neural networks.* Macmillan College Publ., 1994.

[13] K. Hornik, M. Stincombe, and H. White. Neural networks are universal approximators. *Neural Networks*, 2:359–366, 1990.

[14] K. Levenberg. A method for the solution of certain problems in least squares. *Quart. Appl. Math.*, 2:164–168, 1944.

[15] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math.*, 11:431–441, 1963.

[16] D. Nguyen and B. Widrow. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In *IJCNN 1990, San Diego*, volume 3, pages 21–26, 1990.

[17] A. J. Owens and D. L. Filkin. Efficient training of the backpropagation network by solving a system of stiff ordinary differential equations. In *IJCNN 1989, San Diego*, volume 2, pages 381–386, 1989.

[18] W. H. Press. *Numerical recipes in C.* Cambridge Univ. Press, 1995.

[19] W. Schiffmann, M. Joost, and R. Werner. Optimization of the backpropagation algorithm for training multilayer perceptrons. Technical report, Institute of Computer Science, University of Koblenz, ftp://128.146.8.52/pub/neuroprose/schiff.bp_speedup.ps.Z, 1993.

[20] C. W. Therrien. *Decision, estimation and classification.* Wiley, 1989.