MLP in Layer-Wise Form with Applications to Weight Decay

Tommi Kärkkäinen
tka@mit.jyu.fi
Department of Mathematical Information Technology, University of Jyväskylä, P.O.Box 35 (Agora), FIN-40351 Jyväskylä, Finland

A simple and general calculus for the sensitivity analysis of a feedforward MLP network in a layer-wise form is presented. Based on the local optimality conditions, some consequences for the least-means-squares learning problem are stated and further discussed. Numerical experiments with formulation and comparison of different weight decay techniques are included.

1 Introduction

There are many assumptions and intrinsic conditions behind well-known computational techniques that are sometimes lost from a practitioner. In this work, we try to enlighten some of the issues related to multilayered perceptron networks (MLPs). The point of view here is mainly based on the theory and practice of optimization, with special emphasis on sensitivity analysis (computation of gradient), scale balancing, convexity, and smoothness.

First, we propose a simplified way to derive the error backpropagation formulas for the MLP training in a layer-wise form. The basic advantage of the layer-wise formalism is that the optimality system is presented in a compact form that can be readily exploited in an efficient computer realization. Moreover, due to the clear description of the optimality conditions, we are able to derive some consequences and interpretations concerning the final structure of trained network. These results have direct applications to different weight decay techniques, which are presented and tested through numerical experiments.

In section 2, we introduce the algebraic formalism and the original learning problem. In section 3, we compute the optimality conditions for the network learning and derive and discuss some of their consequences. Finally, in section 4 we present numerical experiments for studying different regularization techniques and make some observations based on the computational results.

2 Preliminaries

Action of the multilayered perceptron in a layer-wise form is given by (Rohas, 1996; Hagan & Menhaj, 1994)

$$\mathbf{o}^0 = \mathbf{x}, \quad \mathbf{o}^l = \mathcal{F}^l(\mathbf{W}^l \hat{\mathbf{o}}^{(l-1)}) \quad \text{for } l = 1, \ldots, L. \tag{2.1}$$

We have placed the layer number (starting from zero for the input) as an upper index. By $\hat{}$ we indicate the addition of bias terms, and $\mathcal{F}^l(\cdot)$ denotes the usual componentwise activation on the lth level. The dimensions of the weight matrices are given by $\dim(\mathbf{W}^l) = n_l \times (n_{l-1} + 1)$, $l = 1, \ldots, L$, where n_0 is the length of an input vector $\mathbf{x}$, n_L the length of the output vector $\mathbf{o}^L$, and $n_l, 0 < l < L$, determine the sizes (number of neurons) of the hidden layers.

We notice that the activation of m neurons can be equivalently represented by using a diagonal function matrix $\mathcal{F} = \mathcal{F}(\cdot) = \text{Diag}\{f_i(\cdot)\}_{i=1}^m$ of the form

$$\mathcal{F} = \begin{bmatrix} f_1(\cdot) & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & f_m(\cdot) \end{bmatrix}. \tag{2.2}$$

A function matrix supplied with the natural way to define the matrix vector product $\mathbf{y} = \mathcal{F}(\mathbf{v}) \Leftrightarrow y_i = \sum_{j=1}^m f_{ij}(v_j)$ yields the usual activation approach. However, any enlargement of the function matrix to contain nondiagonal function entries as well offers interesting possibilities for generalizing the basic MLP architecture. This topic is out of the scope of the work presented here, although both the sensitivity analysis and its consequences in what follows remain valid in this case too.

Using a given training data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$, $\mathbf{x}_i \in \mathbb{R}^{n_0}$ and $\mathbf{y}_i \in \mathbb{R}^{n_L}$, the unknown weight matrices $\{\mathbf{W}^l\}_{l=1}^L$ are determined as a solution of the optimization problem

$$\min_{\{\mathbf{W}^l\}_{l=1}^L} \mathcal{J}(\{\mathbf{W}^l\}), \tag{2.3}$$

where

$$\mathcal{J}(\{\mathbf{W}^l\}) = \frac{1}{2N} \sum_{i=1}^N \|\mathcal{N}(\{\mathbf{W}^l\})(\mathbf{x}_i) - \mathbf{y}_i\|^2 \tag{2.4}$$

is the least-mean-squares (LMS) cost functional for a simplified architecture of MLP containing only a linear transformation in the final layer. That is, $f_i^L(x) = x$ for all $1 \le i \le n_L$, and $\mathbf{o}_i^L = \mathcal{N}(\{\mathbf{W}^l\})(\mathbf{x}_i) = \mathbf{W}^L \hat{\mathbf{o}}_i^{(L-1)}$. The relation between this choice and the original form, equation 2.1, will be considered in sections 3.3 and 3.4. In equation 2.4, $\|\cdot\|$ denotes the usual Euclidian norm, which is induced by the l_2 inner product $(\mathbf{v}, \mathbf{w}) = \mathbf{w}^T \mathbf{v}$.

Preprocessing of the training data $\{\mathbf{x}_i, \mathbf{y}_i\}$ has been extensively considered, for example, by LeCun, Bottous, Orr, and Müller (1998). From the optimization point of view, an essential property of any problem is to have all unknowns on the same scale, because gradient with components of different orders of magnitude can yield nonbalanced updates in the training algorithm (Nocedal & Wright, 1999). This usually leads to nonconvergence or large deviation of weights, decreasing the fault tolerance of the trained network (Cavalieri & Mirabella, 1999). Therefore, we require that all activation functions $\{\mathcal{F}^l\}$ have the same range, and all of the training data are prescaled into this range. Then all layers of the network treat vectors with components of the same order of magnitude. Notice, however, that using tanh activation and the proposed prescaling, the average of minimum and maximum values of each feature in $\{\mathbf{x}_i\}$ is transformed to zero. Then all records containing the zero value become insensitive to the corresponding column in the weight matrix $\mathbf{W}^1$. Hence, for features having symmetric and peak-like distribution, the prescaling may result in the nonuniqueness of $\mathbf{W}^1$ (and subsequent matrices $\mathbf{W}^l$, $1 < l \le L$). Naturally, one can study the existence of such problems through the distribution (histogram) of individual prescaled features.

3 Sensitivity Analysis

An essential part of any textbook on neural networks consists of the derivation of error-backpropagation formulas for network training (Bishop, 1995; Reed & Marks, 1999; Rohas, 1996). What makes such sensitivity analysis messy is the consecutive application of the chain rule in an index jungle. Here, we describe another approach that uses the Lagrangian treatment of equality constraint optimization problems (Bertsekas, 1982) and circumvents these technicalities, allowing one to derive the necessary optimality conditions in a straightforward way. The idea behind our treatment is to work on the same level of abstraction, layer-wise form, that is used for the initial description of MLP.

There are many techniques for solving optimization problems like equation 2.3 without the need of precise formulas for the derivatives. One class of techniques consists of the so-called gradient-free optimization methods as presented, for example, in Osman and Kelly (1996) and Bazaraa, Sherali, and Shetty (1993). The computational efficiency of these approaches, however, cannot compete with methods using gradient information during the search procedure. Another interesting possibility for avoiding explicit sensitivity analysis is to use automatic differentiation (AD), which enables the computation of derivatives as a (user-hidden) by-product of cost function evaluation (Griewank, 2000). AD allows fast and straightforward development and simulation of different architectures and learning problems for MLPs, but at the expense of increased storage requirements and computing time compared to the fully exploited analytical approach (see the discussion after theorem 2 in section 3.2). Moreover, clear and precise description of optimality conditions is necessary to be able to analyze the properties of trained networks (cf. sections 3.3 and 3.4).

3.1 MLP with One Hidden Layer. To simplify the presentation, we start with MLP with only one hidden layer. Then any local solution $(\mathbf{W}^{1^*}, \mathbf{W}^{2^*})$ of the minimization problem, equation 2.3, is characterized by the conditions

$$\nabla_{(\mathbf{W}^1,\mathbf{W}^2)}\mathcal{J}(\mathbf{W}^{1^*}, \mathbf{W}^{2^*}) = \begin{bmatrix}\nabla_{\mathbf{W}^1}\mathcal{J}(\mathbf{W}^{1^*}, \mathbf{W}^{2^*})\\ \nabla_{\mathbf{W}^2}\mathcal{J}(\mathbf{W}^{1^*}, \mathbf{W}^{2^*})\end{bmatrix} = \begin{bmatrix}\mathbf{O}\\ \mathbf{O}\end{bmatrix}. \tag{3.1}$$

Here, $\nabla_{\mathbf{W}^l}\mathcal{J} = \left(\frac{\partial \mathcal{J}}{\partial w^l_{ij}}\right)_{i,j}$, $l = 1, 2$, are given in a similar matrix form as the unknown weight matrices. Hence, our next task is to derive the derivatives with regard to these data structures (cf. appendix B in Diamantaras & Kung, 1996; Hagan & Menhaj, 1994). For this analysis, we presuppose that all activation functions in the function matrix $\mathcal{F} = \mathcal{F}^1$ are differentiable.

We start the derivation by stating some simple lemmas for which the proofs are given in appendix 4.2. We note that the proposed approach is not restricted to the LMS error function. For other differentiable cost functionals like softmax and cross-entropy, one can use exactly the same technique for simple derivation of the necessary optimality conditions in a layer-wise form.

Lemma 1. *Let* $\mathbf{v} \in \mathbb{R}^{m_1}$ *and* $\mathbf{y} \in \mathbb{R}^{m_2}$ *be given vectors. The gradient matrix* $\nabla_{\mathbf{W}} J(\mathbf{W}) \in \mathbb{R}^{m_2\times m_1}$ *for the functional* $J(\mathbf{W}) = \frac{1}{2}\,\|\mathbf{W}\,\mathbf{v} - \mathbf{y}\|^2$ *is of the form*

$$\nabla_{\mathbf{W}} J(\mathbf{W}) = [\mathbf{W}\,\mathbf{v} - \mathbf{y}]\,\mathbf{v}^T.$$

Lemma 2. *Let* $\mathbf{W} \in \mathbb{R}^{m_2\times m_1}$ *be a given matrix,* $\mathbf{y} \in \mathbb{R}^{m_2}$ *a given vector, and* $\mathcal{F} = \mathrm{Diag}\{f_i(\cdot)\}_{i=1}^{m_1}$ *a given diagonal function matrix. The gradient vector* $\nabla_{\mathbf{u}} J(\mathbf{u}) \in \mathbb{R}^{m_1}$ *for the functional* $J(\mathbf{u}) = \frac{1}{2}\,\|\mathbf{W}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}\|^2$ *reads as*

$$\nabla_{\mathbf{u}} J(\mathbf{u}) = \left(\mathbf{W}\,\mathcal{F}'(\mathbf{u})\right)^T [\mathbf{W}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}] = \mathrm{Diag}\{\mathcal{F}'(\mathbf{u})\}\,\mathbf{W}^T\,[\mathbf{W}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}].$$

Lemma 3. *Let* $\bar{\mathbf{W}} \in \mathbb{R}^{m_2\times m_1}$ *be a given matrix,* $\mathcal{F} = \mathrm{Diag}\{f_i(\cdot)\}_{i=1}^{m_1}$ *a given diagonal function matrix, and* $\mathbf{v} \in \mathbb{R}^{m_0}$, $\mathbf{y} \in \mathbb{R}^{m_2}$ *given vectors. The gradient matrix* $\nabla_{\mathbf{W}} J(\mathbf{W}) \in \mathbb{R}^{m_1\times m_0}$ *for the functional* $J(\mathbf{W}) = \frac{1}{2}\,\|\bar{\mathbf{W}}\,\mathcal{F}(\mathbf{W}\mathbf{v}) - \mathbf{y}\|^2$ *is of the form*

$$\nabla_{\mathbf{W}} J(\mathbf{W}) = \mathrm{Diag}\{\mathcal{F}'(\mathbf{W}\mathbf{v})\}\,\bar{\mathbf{W}}^T\,[\bar{\mathbf{W}}\,\mathcal{F}(\mathbf{W}\mathbf{v}) - \mathbf{y}]\,\mathbf{v}^T.$$

Now we are ready to state the actual result for the perceptron with one hidden layer.

Theorem 1. *Gradient matrices* $\nabla_{\mathbf{W}^2}\mathcal{J}(\mathbf{W}^1, \mathbf{W}^2)$ *and* $\nabla_{\mathbf{W}^1}\mathcal{J}(\mathbf{W}^1, \mathbf{W}^2)$ *for the cost functional, equation 2.4, are of the form*

$$\begin{aligned}\nabla_{\mathbf{W}^2}\mathcal{J}(\mathbf{W}^1, \mathbf{W}^2) &= \frac{1}{N}\sum_{i=1}^{N}[\mathbf{W}^2\,\widehat{\mathcal{F}}(\mathbf{W}^1\,\hat{\mathbf{x}}_i) - \mathbf{y}_i]\,[\widehat{\mathcal{F}}(\mathbf{W}^1\,\hat{\mathbf{x}}_i)]^T \qquad i.\\ &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{e}_i\,[\widehat{\mathcal{F}}(\mathbf{W}^1\,\hat{\mathbf{x}}_i)]^T,\\ \nabla_{\mathbf{W}^1}\mathcal{J}(\mathbf{W}^1, \mathbf{W}^2) &= \frac{1}{N}\sum_{i=1}^{N}\mathrm{Diag}\{\mathcal{F}'(\mathbf{W}^1\hat{\mathbf{x}}_i)\}(\mathbf{W}^2_1)^T[\mathbf{W}^2\widehat{\mathcal{F}}(\mathbf{W}^1\hat{\mathbf{x}}_i) - \mathbf{y}_i]\hat{\mathbf{x}}_i^T \qquad ii.\\ &= \frac{1}{N}\sum_{i=1}^{N}\mathrm{Diag}\{\mathcal{F}'(\mathbf{W}^1\,\hat{\mathbf{x}}_i)\}\,(\mathbf{W}^2_1)^T\,\mathbf{e}_i\,\hat{\mathbf{x}}_i^T.\end{aligned}$$

In formula ii, $\mathbf{W}^2_1$ *is the submatrix* $\left(\mathbf{W}^2\right)_{i,j}$, $i = 1, \ldots, n_2$, $j = 1, \ldots, n_1$, *which is obtained from* $\mathbf{W}^2$ *by removing the first column* $\mathbf{W}^2_0$ *containing the bias nodes.*

Proof. Formula i is a direct consequence of lemma 1. Moreover, due to the definition of the extension operator $\widehat{\ }$ we have, for all $1 \le i \le N$,

$$\begin{aligned}\mathbf{W}^2\,\widehat{\mathcal{F}}(\mathbf{W}^1\,\hat{\mathbf{x}}_i) - \mathbf{y}_i &= [\mathbf{W}^2_0\mathbf{W}^2_1]\begin{bmatrix}1\\ \mathcal{F}(\mathbf{W}^1\,\hat{\mathbf{x}}_i)\end{bmatrix} - \mathbf{y}_i\\ &= \mathbf{W}^2_0 + \mathbf{W}^2_1\,\mathcal{F}(\mathbf{W}^1\,\hat{\mathbf{x}}_i) - \mathbf{y}_i.\end{aligned} \tag{3.2}$$

Using equation 3.2 and lemma 3 componentwise for the cost functional, equation 2.4, shows formula ii and ends the proof.

3.2 MLP with Several Hidden Layers. Next, we generalize the previous analysis to the case of several hidden layers.

Lemma 4. *Let* $\tilde{\mathbf{W}} \in \mathbb{R}^{m_3\times m_2}$ *and* $\bar{\mathbf{W}} \in \mathbb{R}^{m_2\times m_1}$ *be given matrices,* $\tilde{\mathcal{F}} = \mathrm{Diag}\{\tilde{f}_i(\cdot)\}_{i=1}^{m_2}$ *and* $\bar{\mathcal{F}} = \mathrm{Diag}\{\bar{f}_i(\cdot)\}_{i=1}^{m_1}$ *given diagonal function matrices, and* $\mathbf{v} \in \mathbb{R}^{m_0}$, $\mathbf{y} \in \mathbb{R}^{m_3}$ *given vectors. The gradient matrix* $\nabla_{\mathbf{W}} J(\mathbf{W}) \in \mathbb{R}^{m_1\times m_0}$ *for the functional* $J(\mathbf{W}) = \frac{1}{2}\,\|\tilde{\mathbf{W}}\,\tilde{\mathcal{F}}(\bar{\mathbf{W}}\,\bar{\mathcal{F}}(\mathbf{W}\mathbf{v})) - \mathbf{y}\|^2$ *is of the form*

$$\begin{aligned}\nabla_{\mathbf{W}} J(\mathbf{W}) = \mathrm{Diag}\{\bar{\mathcal{F}}'(\mathbf{W}\mathbf{v})\}\,\bar{\mathbf{W}}^T\,\mathrm{Diag}\{\tilde{\mathcal{F}}'(\bar{\mathbf{W}}\,\bar{\mathcal{F}}(\mathbf{W}\mathbf{v}))\}\\ \tilde{\mathbf{W}}^T\,[\tilde{\mathbf{W}}\,\tilde{\mathcal{F}}(\bar{\mathbf{W}}\,\bar{\mathcal{F}}(\mathbf{W}\mathbf{v})) - \mathbf{y}]\,\mathbf{v}^T.\end{aligned}$$

Theorem 2. *Gradient-matrices $\nabla_{\mathbf{W}^l}\mathcal{J}(\{\mathbf{W}^l\})$, $l = L, \ldots, 1$, for the cost functional equation 2.4 read as*

$$\nabla_{\mathbf{W}^l}\mathcal{J}(\{\mathbf{W}^l\}) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{d}_i^l\,[\hat{\mathbf{o}}_i^{(l-1)}]^T,$$

where

$$\mathbf{d}_i^L = \mathbf{e}_i = \mathbf{W}^L\hat{\mathbf{o}}_i^{(L-1)} - \mathbf{y}_i, \tag{3.3}$$

$$\mathbf{d}_i^l = \mathrm{Diag}\{(\mathcal{F}^l)'(\mathbf{W}^l\,\hat{\mathbf{o}}_i^{(l-1)})\}\,(\mathbf{W}_1^{(l+1)})^T\,\mathbf{d}_i^{(l+1)}. \tag{3.4}$$

Proof. Apply lemma 4 inductively.

The definition of $\mathbf{d}_i^l$ contains the backpropagation of the output error in a layer-wise form. The compact presentation of the optimality system can be readily exploited in the implementation. More precisely, computation of the activation derivatives $(\mathcal{F}^l)'(\mathbf{W}^l\,\hat{\mathbf{o}}_i^{(l-1)})$ can, when using the usual logistic sigmoid or hyperbolic tangent functions, be realized using layer outputs $\mathbf{o}_i^l$ without additional storage. Hence, in the forward loop, it is enough to store the outputs of different layers for the backward gradient loop, where these same vectors can be overwritten by $\mathbf{d}_i^l$ when the whole operation in equation 3.4 is implemented using a single loop. In modern workstations, such combination of operations yielding minimal amount of loops through the memory can decrease the computing time significantly (Kärkkäinen & Toivanen, 2001).

3.3 Some Corollaries. Next, we derive some corollaries of the layer-wise optimality conditions. All of these results follow from having a linear transformation with biases in the final layer of MLP. Moreover, whether one applies on-line or batch training makes no difference if the optimization problem 2.3 is solved with high accuracy, because then both methods yield $\nabla_{\mathbf{W}^l}\mathcal{J}(\{\mathbf{W}^{l^*}\}) \simeq \mathbf{O}$ for the local solution $\{\mathbf{W}^{l^*}\}$.

Corollary 1. *For locally optimal MLP network satisfying the conditions in theorem 2:*

i. The average error $\frac{1}{N}\sum_{i=1}^{N}\mathbf{e}_i^*$ is zero.

ii. Correlation between the error vectors and the action of layer $L-1$ is zero.

Proof. The optimality condition $\nabla_{\mathbf{W}^L}\mathcal{J}(\{\mathbf{W}^{l^*}\}) = \frac{1}{N}\sum_{i=1}^{N}\mathbf{e}_i^*\,[\hat{\mathbf{o}}_i^{(L-1)}]^T = \mathbf{O}$ (with the abbreviation $\hat{\mathbf{o}}_i^{(L-1)} = \hat{\mathbf{o}}_i^{(L-1)^*}$) in theorem 2 can be written in the nonextended form as

$$\frac{1}{N}\sum_{i=1}^{N}\mathbf{e}_i^*\,[1\;(\mathbf{o}_i^{(L-1)})^T] = \mathbf{O}. \tag{3.5}$$

By taking the transpose of this, we obtain

$$\frac{1}{N}\sum_{i=1}^{N}\begin{bmatrix}1\\ \mathbf{o}_i^{(L-1)}\end{bmatrix}(\mathbf{e}_i^*)^T = \frac{1}{N}\sum_{i=1}^{N}\begin{bmatrix}(\mathbf{e}_i^*)^T\\ \mathbf{o}_i^{(L-1)}\,(\mathbf{e}_i^*)^T\end{bmatrix} = \begin{bmatrix}\mathbf{O}\\ \mathbf{O}\end{bmatrix}. \tag{3.6}$$

This readily shows both results.

In the next corollary, we recall some conditions concerning the final weight matrix $\mathbf{W}^L$. We note that the assumption on nonsingularity below can be relaxed to pseudo-invertibility, which has been widely used to generate new training algorithms for the MLP networks (Di Martino, Fanelli, & Protasi, 1996; Wang & Chen, 1996).

Corollary 2. *If the autocorrelation matrix $\mathbf{A} = \frac{1}{N}\sum_{i=1}^{N}\widehat{\mathbf{v}}_i\widehat{\mathbf{v}}_i^T$ of the (extended) final layer inputs $\mathbf{v}_i = \mathbf{o}_i^{(L-1)}$ is nonsingular, $\mathbf{W}^L$ can be recovered from the formula*

$$\mathbf{W}^L = \mathbf{B}\,\mathbf{A}^{-1} \quad \textit{for} \;\; \mathbf{B} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{y}_i\,\widehat{\mathbf{v}}_i^T. \tag{3.7}$$

Furthermore, if $\mathbf{A}^*$ for a minimizer $\{\mathbf{W}^{l^*}\}_{l=1}^{L}$ of equation 2.4 is nonsingular, then $\mathbf{W}^{L^*}$ is unique.

Proof. $\mathbf{W}^L$ satisfying the system

$$\frac{1}{N}\sum_{i=1}^{N}[\mathbf{W}^L\,\hat{\mathbf{o}}_i^{(L-1)} - \tilde{\mathbf{y}}_i]\,[\hat{\mathbf{o}}_i^{(L-1)}]^T = \mathbf{O} \tag{3.8}$$

is independent of i. Hence, equation 3.8 can be written as

$$\mathbf{W}^L\,\frac{1}{N}\sum_{i=1}^{N}\widehat{\mathbf{v}}_i\,\widehat{\mathbf{v}}_i^T = \frac{1}{N}\sum_{i=1}^{N}\tilde{\mathbf{y}}_i\,\widehat{\mathbf{v}}_i^T \quad \Leftrightarrow \quad \mathbf{W}^L\,\mathbf{A} = \mathbf{B}. \tag{3.9}$$

Multiplying both sides from right with $\mathbf{A}^{-1}$ gives equation 3.7 and ends the proof.

Let us add one further observation concerning the above result. Because $\dim(\mathrm{span}(\widehat{\mathbf{v}}_i)) = 1$, each matrix $\widehat{\mathbf{v}}_i\widehat{\mathbf{v}}_i^T$ has only one nonzero eigenvalue $\lambda = \widehat{\mathbf{v}}_i^T\widehat{\mathbf{v}}_i$ with the corresponding eigenvector $\widehat{\mathbf{v}}_i$. This means that $1 \le \dim(\mathrm{span}(\mathbf{A})) \le \min(N, n_{L-1}+1)$. Hence, if $N < n_{L-1}+1$, then $\mathbf{A}$ must be singular and $\mathbf{W}^{L^*}$ cannot be unique. On the other hand, if $N > n_{L-1}+1$, then all vectors $\widehat{\mathbf{v}}_i$ cannot be linearly independent. This shows that there exists a relation between the amount of learning data and an appropriate size of the last hidden layer for the LMS learning problem.

3.4 Some Consequences. We conclude this section by giving a list of observations and comments based on the previous results.

3.4.1 Final Layer with or Without Activation. If MLP also contains the final layer activation $\mathcal{F}^L(\mathbf{W}^L\hat{\mathbf{o}}_i^{(L-1)})$, then it follows from lemma 3 (choose $\bar{\mathbf{W}} = \mathbf{I}$) that equation 3.3 replaced with

$$\mathbf{d}_i^L = \mathrm{Diag}\{(\mathcal{F}^L)'(\mathbf{W}^L\,\hat{\mathbf{o}}_i^{(L-1)})\}\,\mathbf{e}_i = \mathbf{D}_i\,\mathbf{e}_i \tag{3.10}$$

gives the corresponding sensitivity with regard to $\mathbf{W}^L$. In this case, we have in corollary 1 instead of equation 3.6,

$$\frac{1}{N}\sum_{i=1}^{N}\begin{bmatrix}(\mathbf{e}_i^*)^T\\ \mathbf{o}_i^{(L-1)}(\mathbf{e}_i^*)^T\end{bmatrix}\mathbf{D}_i = \begin{bmatrix}\mathbf{O}\\ \mathbf{O}\end{bmatrix}.$$

Hence, the two formulations with and without activating the final layer yield locally the same result for the zero-residual problem $\mathbf{e}_i^* = 0$ for all $1 \le i \le N$. This slightly generalizes the corresponding result derived by Moody and Antsaklis (1996), where bijectivity of the final activation $\mathcal{F}^L$ was also assumed.

The use of 0-1 coding for the desired output vectors in classification enforces the weight vector of 1-neuron to the so-called saturation area of a sigmoidal activation where the derivative is nearly zero (Vitela & Reifman, 1997). For this reason, the error function with the final layer activation has a nearly flat region around such points, whereas the final linear layer has no such problems. This is also evident from equation 3.10, which holds independently on $\mathbf{e}_i^*$ and $\mathbf{o}_i^{(L-1)}$ for $\mathbf{D}_i = \mathbf{O}$. Indeed, the tests reported by Gorse, Shepherd, and Taylor (1997) (see also Figure 8.7 in Reed & Marks, 1999; de Villiers & Barnard, 1992) suggest that the network with sigmoid in the final layer has more local minima than when using a linear final layer. Furthermore, Japkowicz, Hanson, and Gluck (2000) pointed out that the reconstruction error surface for a sigmoidal autoassociator consists of multiple local valleys on the contrary to the linear one.

3.4.2 The Basic Interpretation. We recall that the transformation before the last linear layer with biases had no influence whatsoever on the previous formal consequences. This suggests the following interpretation of the MLP action with the proposed architecture: nonlinear hidden layers produce the universal approximation capability of MLP, while the final linear layer compensates the hidden action with the desired output in an uncorrelated and error-averaging manner (cf. Japkowicz et al., 2000).

3.4.3 The Basic Consequence. The simplest model of noise in regression is to assume that given targets are generated by

$$\mathbf{y}_i = \phi(\mathbf{x}_i) + \varepsilon_i, \tag{3.11}$$

where $\phi(\mathbf{x})$ is the unknown stationary function and ε_i's are sampled from an underlying noise process. For equation 3.11, it follows from corollary 1i that a locally optimal MLP network $\mathcal{N}(\{\mathbf{W}^{l^*}\})$ satisfies

$$\frac{1}{N}\sum_{i=1}^{N}\Big[\mathcal{N}(\{\mathbf{W}^{l^*}\})(\mathbf{x}_i) - \phi(\mathbf{x}_i)\Big] = \frac{1}{N}\sum_{i=1}^{N}\varepsilon_i. \tag{3.12}$$

Hence, this shows that every $\mathcal{N}(\{\mathbf{W}^{l^*}\})$ treats optimally gaussian noise with zero mean (and enough samples). This result is not valid for other error functions or with final layer activation (except in the impractical zero-residual case), but it remains valid for networks with input enlargements (e.g., Flake, 1998), output enlargements (Caruna, 1998), hidden layer modifications (e.g., the linearly augmented feedforward network as proposed by van der Smagt & Hirzinger, 1998), and convex combinations of different locally optimal networks (averaged ensemble, e.g., Liu & Yao, 1999; Horn, Naftaly, & Intrator, 1998) for all cases when the final bias is left untouched in the architecture.

3.4.4 Implicit Prior and Its Modification. Many authors writing on ANNs note that changing the prior frequency of different samples in the training data to favor the rare ones may improve the performance of the obtained network (e.g., LeCun et al., 1998; Yaeger, Webb, & Lyon, 1998). Corollary 1i gives a precise explanation how such a modification alters the final result. For stochastic on-line learning, the prior frequency can be altered by controlling the feeding of examples, but for batch learning, one needs an explicit change in the error function for this purpose. For example, consider the classification problem with learning data from K different classes $\{C_k\}_{k=1}^K$ so that $N = \sum_{k=1}^K N_k$, where N_k denotes the number of samples from the class C_k. To have an equal prior probability $\frac{1}{K}$ for all classes in the classifier (instead of N_k/N for the kth class), one needs to adjust the LMS cost functional to

$$\mathcal{J}(\{\mathbf{W}^l\}_{l=1}^L) = \sum_{k=1}^{K}\frac{1}{2KN_k}\sum_{i\in C_k}\|\mathbf{W}^L\hat{\mathbf{o}}_i^{(L-1)} - \mathbf{y}_i\|^2.$$

Hence, using nonconstant weighting of the learning data in the cost functional incorporates prior information into the learning problem. For example, a time-series prediction could be based on larger weighting of the more recent samples, especially if one tries to simulate a slowly varying dynamical process (Park, El-Sharkawi, & Marks, 1991). It is also straightforward to improve the LMS learning problem when more knowledge on the variance of the output data is available (MacKay, 1992; van de Laar & Heskes, 1999). Finally, a locally weighted linear regression also adaptable for MLP learning is introduced by Schaal and Atkeson (1998).

3.4.5 Relation to Early Stopping. A popular way to try to improve the generalization power of MLP is to use early stopping (cutting, termination) of the learning algorithm (Bishop, 1995; Tetko & Villa, 1997; Prechelt, 1998). In most cases, early stopping (ES) is due to a cross-validation technique, but basically it also happens every time when a learning problem is solved inexactly, for example, because of a fixed number of epochs or a bad learning rate.

Formulation of general and robust conditions to stop different optimization algorithms prematurely is problematic and usually leads to a very large number of tests for validating different networks (Gupta & Lam, 1998). Moreover, Cataltepe, Abu-Mostafa, and Magdon-Ismail (1999) showed that if models with the same training error are chosen with equal probability, then the lowest generalization error is obtained by choosing the model corresponding to the training error minimum. This result is valid globally for linear models and locally, around the training error minimum, for nonlinear models.

The success of ES is sometimes argued as due to producing smoother results when the network is initialized with small random numbers in the almost linear region of a sigmoidal activation function. This is certainly vague because different optimization methods follow different search paths, and nothing guarantees that even an overly hasty stopped training algorithm produces smooth networks. Hence, we believe that the role of ES is more related to equation 3.12, which for an inexact solution is not valid. Namely, for a nongaussian error distribution, ES may decrease the significance of heavy error tails and, due to this fact, produce a result that generalizes better. Especially in this case, ES and weight decay (WD) are not alternatives to each other, because ES may improve the learning problem and WD can favor simpler models during learning. Notice, however, that because the usual bias-variance decomposition (Geman, Bienenstock, & Doursat, 1992) of the expected generalization error is also based on the least-squares estimation, the least-squares-based cross-validation technique may not be appropriate for a nongaussian error in the validation set. Changing the underlying gaussian assumption on noise should instead lead to the derivation of new cost functionals for the learning problem (Chen & Jain, 1994; Liano, 1996).

4 Numerical Results

Here, we describe numerical experiments based on the proposed techniques. All experiments are performed on an HP9000/J280 workstation (180 MHz PA8000 CPU), and the implementation is based on F77 (optimization and MLP realization) and Matlab for data pre- and postprocessing. (More comprehensive coverage of the computed examples is presented in Kärkkäinen, 2000.)

As an optimization software, we apply the limited memory quasi-Newton subroutine L-BFGS (Byrd, Lu, & Nocedal, 1995), which uses a sparse approximation of the BFGS-formula-based inverse of the Hessian matrix and is intended for solving large nonlinear optimization problems efficiently. As a stopping criterion for the optimization, we use

$$\frac{\mathcal{J}^k - \mathcal{J}^{k+1}}{\max\left\{|\mathcal{J}^k|, |\mathcal{J}^{k+1}|, 1\right\}} \le 10^6 * \mathit{epsmch},$$

where *epsmch* is the machine epsilon ($\sim 10^{-16}$ in the present case). This choice reflects our intention to solve the optimization problem with (unnecessary) high precision for test purposes. The main ingredient in the L-BFGS software is that due to the limited-memory Hessian update, the computational complexity is only $\mathcal{O}(n)$, where n is the total amount of unknowns. For ordinary quasi-Newton methods, the $\mathcal{O}(n^2)$ consumption due to full Hessian has been one of the main reasons preventing the application of these methods for learning problems with larger networks.

There exists a large variety of different tests comparing backpropagation (gradient descent with constant learning rate), conjugate gradient, and second-order methods (Gauss-Newton, Levenberg-Marquart, Hessian approximation, and quasi-Newton) for MLP training (e.g., Bishop, 1995; Hagan & Menhaj, 1994; Magoulas, Vrahatis, & Androulakis, 1999; McKeown, Stella, & Hall, 1997; Wang & Lin, 1998). The difficulty of drawing reliable conclusions between the quality of different methods is that in addition to how to solve it (training method), as (or even more) important is also to consider what to solve (learning problem) to connect our approach to an application represented by a finite set of samples. According to Gorse et al. (1997), a quasi-Newton method can survey a larger amount of local minima than the BP and CG methods, but to truly enforce search through different minima when using a local gradient-based method, one must either start the training using multiple initial configurations or apply some global optimization strategy as on outer iteration.

In the numerical experiments, we consider only simple examples, because the emphasis here is on testing the algorithms and different formulations rather than on complex applications of MLPs. Therefore, we also restrict ourselves to the perceptron with only one hidden layer. Concerning the discussion between one or more hidden layers, we refer to Reed and Marks (1999) and Tamura and Tateishi (1997). Finally, notice that more complex (nonconvex) optimization problems with an increased number of local minima must be solved when training a network with several hidden layers.

In order to generate less regular nonlinear transformations without affecting the scale of weights when the hidden layer is enlarged, we choose k-tanh functions of the form

$$t_k(a) = \frac{2}{1+\exp(-2\,k\,a)} - 1, \quad k = 1, \ldots, n_1, \tag{4.1}$$

to activate the hidden neurons (McLean, Bandar, & O'Shea, 1998). Although we at the same time introduce some kind of ordering for the hidden layer by using different activation functions for each neuron, the symmetry problem related to the hidden neurons (e.g., Bishop, 1995) remains, as can be seen by a simple rescaling argument. Use of an even more general mixture of different activation functions in the hidden layer is suggested by Zhang and Morris (1998). Finally, by increasing the nonsmoothness of the hidden activation functions, and hence the whole MLP mapping, decreases the regularity of the learning problem and thus also the convergence rate of first- and second-order optimization methods (Nocedal & Wright, 1999).

4.1 Approximation of Noisy Function.

Example 1. Reconstruction of function $f(x) = \sin(2\pi x)$, $x \in I = [0, 2\pi]$, which is corrupted with normally distributed (quasi-)random noise. The input data $\{x_i\}$ result from the uniform discretization of the interval I with the step size $h = 0.1$. Points in the output data are taken as $y_i = f(x_i) + \delta\,\varepsilon_i$ for $\delta = 0.3$ and $\varepsilon_i \in \mathcal{N}(0, 1)$. Altogether, we have in this example $N = 63$ and $n_2 = n_0 = 1$.

In the following experiments we have solved the optimization problem, equation 2.3, with prescaled learning data into $[-1, 1]$ starting from 10 random initial guesses from the range $(-1, 1)$ for the weight matrices $(\mathbf{W}^1, \mathbf{W}^2)$. For an overview and study of different initialization techniques we refer to Thimm and Fiesler (1997) and LeCun et al. (1998).

Let us make some comments based on Table 1:

Local minima: Even for the smallest network ($n_1 = 2$) and especially for larger ones, there exist a lot of local minima in the optimization problem. Moreover, the local minima are strict in the sense that they correspond to truly different values of the cost functional and not just different representations (symmetries) of the same MLP transformation.

Condition i in Corollary 1: Is valid with a precision related to the stopping criterion.

Table 1: Computational Results in Example 1 Without Regularization.

	$\mathcal{J}^*$			$\lvert\bar{e}^*\rvert$			Its			CPU		
n_1	Minimum	Maximum	Mean	Minimum	Maximum	Mean	Minimum	Maximum	Mean	Minimum	Maximum	Mean
2	41	234	126	0.23	0.52	0.39	0.014 (2)	0.032	0.018	2e-8	4e-6	2e-6
3	120	1079	427	0.42	0.80	0.60	0.013	0.024	0.014	4e-7	2e-5	4e-6
4	153	341	250	0.47	0.60	0.54	0.012 (2)	0.014	0.013	1e-6	8e-6	4e-6
5	0.010	0.013	0.012	4e-7	2e-5	6e-6	237	804	430	0.55	0.76	0.64
6	0.010	0.013	0.011	3e-7	3e-5	8e-6	183	737	466	0.51	0.75	0.66
7	0.0086	0.013	0.010	2e-7	2e-5	7e-6	440	1541	927	0.67	0.95	0.80

Notes: The minimum, maximum, and average mean values correspond to the 10 solutions of the optimization problem for the following quantities: $\mathcal{J}^*$ is the final value of the cost functional. If the minimum or maximum value (with tolerance $\varepsilon = 10^{-6}$) is scored more than once, the number of instances is included in parentheses. $\lvert\bar{e}^*\rvert$ denotes the absolute value of the average output error over the learning data as stated in corollary 1i. Its = total number of function/gradient evaluations (always computed together). CPU = time in seconds for solving the optimization problem.

Efficiency: CPU time is negligible in this small example, but the number of iterations in the optimization varies a lot.

Generalization: Visually (Kärkkäinen, 2000) the best result is obtained using the MLP corresponding to the minimal value of $\mathcal{J}^*$ for $n_1 = 2$. However, MLP corresponding to the maximal value of $\mathcal{J}^*$ for $n_1 = 7$ also gives a good result. This illustrates the difficulty of naming the optimal network even in a simulated example. Moreover, from the large variation of the number of iterations, we conclude that when a fixed number of iterations is taken in the learning algorithm, one has no knowledge on the error between the obtained weights and the true (local) solution of the optimization problem.

4.1.1 Regularization of MLP Using Weight Decay. Weight decay (WD) is a popular technique for pruning an MLP (Goutte & Hansen, 1997; Gupta & Lam, 1998; Hintz-Madsen, Hansen, Larsen, Pederson, & Larsen, 1998; Ormoneit, 1999; Reed & Marks, 1999). In WD, the basic LMS error function is augmented with a WD term $\mathcal{R}(\mathbf{W}^l_{ij})$, which imposes some restriction on the generality (universality) of the MLP transform to prevent overlearning. Here we start with the simplest possible strictly convex form with regard to the unknown weights by considering initially $\mathcal{R}(\mathbf{W}^l_{ij}) = \beta/2 \sum_{l,i,j}(\mathbf{W}^l_{ij})^2$, where β is the WD parameter.

The choice of having only a single coefficient makes sense, because the network inputs and outputs of the hidden layer are enforced in the same range. Moreover, for the gaussian noise with known estimate of variance, β is actually the inverse of the Lagrange multiplier for the corresponding equality or inequality constraint and therefore single-valued (Chambolle & Lions, 1997). In general the "best" value of β is related to both the complexity of the MLP transformation and the (usually unknown) amount of noise contained in the learning data (Scherzer, Engl, & Kunisch, 1993). There exist, however, various techniques for obtaining an effective choice of β (e.g., Bishop, 1995; Rohas, 1996; Rögnvaldsson, 1998).

In addition to strict convexity, some particular reasons for choosing the proposed form of WD with the quadratic penalty function $p(w) = |w|^2$ are:

- This form improves the convexity of the cost functional and therefore makes the learning problem easier to solve.
- The proposed form forces weights in the neighborhood of zero (similarly to prior distributions with zero mean suggested by Neal, 1996), thus further balancing their scale in a gradient-based optimization algorithm (Cavalieri & Mirabella, 1999). The smoothing property is due to the fact that the activation functions $t_k(a)$ are nearly linear around zero, although the size of the linear region is decreasing as k is increasing. Furthermore, first derivatives of the activation functions are most informative around zero, so that the proposed form of WD is also helpful to prevent saturation of weights by enforcing them in the neighborhood of this transient region (Kwon & Cheng, 1996; Vitela & Reifman, 1997).

One drawback of quadratic WD is that it produces weight matrices with groups of small components even if a choice of one large weight instead could be sufficient. This can yield unnecessarily large networks, even if the overlearning can be prevented. On the other hand, this property increases the fault tolerance of the trained network due to the so-called graceful degradation (Reed & Marks, 1999).

Let us comment on some of the difficulties of other forms of WD for individual weights suggested in the literature (Goutte & Hansen, 1997; Gupta & Lam, 1998; Saito & Nakano, 2000):

l^1**–formulation:** Each weight w is regularized using a penalization $p(w) = |w|$. This function is convex but not strictly so. A severe difficulty is that because the derivative of $p(w)$ is multivalued for $w = 0$, the resulting optimization problem is nonsmooth, that is, only subdifferentiable (Mäkelä & Neittaanmäki, 1992). In general, function $|w|^p$ for $1 < p < 2$ belongs only to the Hölder space $C^{1,p-1}$ (Gilbarg & Trudinger, 1983), so that assumptions for convergence of gradient descent (on batch-mode Lipschitz continuity of gradient, for on-line stochastic iteration C^2 continuity), CG (Lipschitz continuity of gradient), and especially quasi-Newton methods (C^2-continuity) are violated (Haykin, 1994; Nocedal & Wright, 1999). As documented, for example, for MLP by Saito and Nakano (2000) and for image restoration by Kärkkäinen, Majava, and Mäkelä (2001), this yields nonconvergence of ordinary training algorithms, when the cost functional does not fulfill the required smoothness assumptions. Furthermore, even if a smoothed counterpart $\sqrt{w^2 + \varepsilon}$ for $\varepsilon > 0$ is introduced, this formulation is either (for small ε) too close to the nonsmooth case so that again the convergence fails, or otherwise (for larger ε), the smoothed formulation differs substantially from the original one. To conclude, for such nonsmooth optimization problems, one needs special algorithms (Kärkkäinen & Majava, 2000a, 2000b; Kärkkäinen et al., 2001). Finally, these same difficulties also concern the so-called robust backpropagation where the error function is defined as $\sum_i \|\mathcal{N}(\{\mathbf{W}^l\})(\mathbf{x}_i) - \mathbf{y}_i\|_{l_1}$ (Kosko, 1992).

Mixed WD: Each weight is regularized using mixed penalization $p(w) = w^2/(1 + w^2)$. This form is nonconvex, producing even more local minima to the optimization problem, and it favors both small and large weights, thus destroying the balance in scales of different weights.

The numerical results reported by Saito and Nakano (2000) emphasize the above difficulties. Whereas the combination of the quasi-Newton optimization algorithm and quadratic WD drastically improved both the convergence of the training algorithm and the generalization performance of the trained MLP, the l_1-penalized problem was not convergent, and the mixed form of WD was very unstable.

However, corollary 1 and the role of bias terms as a shift from the origin raise the question: Which components of the weight matrices $\{\mathbf{W}^l\}_{l=1}^L$ should be regularized and which not? Certainly, if condition i in corollary 1 is required for the resulting MLP, one should exclude the bias terms of the final layer $\mathbf{W}^L$ from WD. Similarly, for both conditions to hold, one should leave out all components of $\mathbf{W}^L$ from WD.

Hence, we apply the quadratic WD only to selected components of the weight-matrices $(\mathbf{W}^1, \mathbf{W}^2)$. We distinguish four cases:

I Regularize all other components except the bias terms $\mathbf{W}_0^2$ in the weight matrix $\mathbf{W}^2$.

II Exclude all components of $\mathbf{W}^2$ from the regularization.

III Exclude all bias terms of $(\mathbf{W}^1, \mathbf{W}^2)$ from the regularization (Holmström, Koistinen, Laaksonen, & Oja, 1997).

IV Exclude all components of $\mathbf{W}^2$ and bias terms of $\mathbf{W}^1$ from the regularization.

Remark. Let us state one further observation concerning the nonregularization of $\mathbf{W}_0^2$. Using the error-average formula $\frac{1}{N}\sum_{i=1}^N \mathbf{e}_i^* = \mathbf{O}$ of corollary 1 and the expression for $\mathbf{e}_i^*$ according to equation 3.2 yields (cf. Bishop, 1995)

$$\mathbf{W}_0^{2,*} = -\frac{1}{N}\sum_{i=1}^{N}[\mathbf{W}_1^{2,*}\,\mathcal{F}(\mathbf{W}^{1,*}\,\hat{\mathbf{x}}_i) - \mathbf{y}_i].$$

Hence, increased coercivity and thereby uniqueness with regard to $(\mathbf{W}_1^2, \mathbf{W}^1)$ immediately affect the uniqueness of the bias $\mathbf{W}_0^2$ as well.

The results corresponding to the above cases are presented in Tables 2 through 5. For all experiments, we have chosen $\beta = 10^{-3}$ according to some prior tests, which also indicated that the results obtained here were not very sensitive to the choice of β.

Let us state some observations based on Tables 1 through 5:

Local minima: For regularization methods I and III, the minimal cost function values are scored more than once when n_1 is small. However, there still exist a lot of local minima for larger networks.

Condition i in corollary 1: Is valid for all regularization methods with a precision related to the stopping criterion.

Efficiency: By means of the number of iterations and the CPU time, regularization methods I and III improved the performance, whereas

Table 2: Computational Results in Example 1 for Regularization I.

	$\mathcal{J}_\beta^*$			$\lvert\bar{e}^*\rvert$			Its			CPU		
n_1	Minimum	Maximum	Mean	Minimum	Maximum	Mean	Minimum	Maximum	Mean	Minimum	Maximum	Mean
2	0.022 (5)	0.025 (5)	0.023	3e-8	4e-6	1e-6	48	84	65	0.25	0.35	0.30
3	0.0195	0.0199 (2)	0.0197	3e-8	5e-6	2e-6	76	149	101	0.33	0.43	0.37
4	0.017	0.019	0.018	1e-7	4e-6	2e-6	88	208	146	0.38	0.51	0.44
5	0.017	0.018	0.017	1e-7	2e-5	7e-6	142	245	188	0.46	0.55	0.50
6	0.016	0.017 (2)	0.017	5e-7	1e-5	6e-6	145	382	244	0.47	0.63	0.55
7	0.016	0.017	0.016	2e-7	2e-5	5e-6	227	395	327	0.57	0.65	0.62

Note: Here and in the sequel, $\mathcal{J}_\beta^*$ refers to the value of the cost functional containing the regularization term.

Table 3: Computational Results in Example 1 for Regularization II.

	$\mathcal{J}_\beta^*$			$\lvert\bar{e}^*\rvert$			Its			CPU		
n_1	Minimum	Maximum	Mean	Minimum	Maximum	Mean	Minimum	Maximum	Mean	Minimum	Maximum	Mean
2	0.015	0.016	0.015	2e-6	8e-5	2e-5	67	554	231	0.33	0.67	0.49
3	0.014	0.015	0.015	2e-7	1e-5	3e-6	131	1009	383	0.44	0.78	0.59
4	0.014	0.015	0.015	8e-8	2e-5	8e-6	257	963	414	0.48	0.79	0.61
5	0.014	0.015 (2)	0.014	2e-6	3e-5	1e-5	329	1283	832	0.61	0.87	0.75
6	0.013	0.014	0.014	3e-6	3e-5	1e-5	607	2402	1331	0.72	1.10	0.88
7	0.013	0.014	0.014	1e-7	1e-5	4e-6	1213	2472	1599	0.86	1.13	0.96

Table 4: Computational Results in Example 1 for Regularization III.

	$\mathcal{J}^*_\beta$			$\lvert e^* \rvert$			Its			CPU		
n_1	Minimum	Maximum	Mean	Minimum	Maximum	Mean	Minimum	Maximum	Mean	Minimum	Maximum	Mean
2	0.022 (5)	0.025 (5)	0.023	1e-7	3e-6	1e-6	49	93	71	0.25	0.36	0.30
3	0.019 (4)	0.025	0.019	3e-8	7e-6	2e-6	73	173	109	0.33	0.48	0.39
4	0.017 (2)	0.019	0.018	3e-7	6e-6	2e-6	87	172	121	0.36	0.48	0.41
5	0.017	0.019	0.017	3e-7	1e-5	4e-6	80	205	141	0.36	0.51	0.45
6	0.016	0.017	0.017	3e-7	7e-6	3e-6	104	199	153	0.40	0.51	0.47
7	0.016	0.018	0.016	2e-6	3e-5	9e-6	118	537	269	0.44	0.70	0.56

Table 5: Computational Results in Example 1 for Regularization IV.

	$\mathcal{J}^*_\beta$			$\lvert e^* \rvert$			Its			CPU		
n_1	Minimum	Maximum	Mean	Minimum	Maximum	Mean	Minimum	Maximum	Mean	Minimum	Maximum	Mean
2	0.015	0.016	0.015	4e-7	2e-5	1e-5	73	913	235	0.31	0.75	0.47
3	0.014	0.016	0.015	9e-7	5e-5	1e-5	110	953	352	0.39	0.77	0.56
4	0.014	0.015	0.015	2e-7	3e-5	9e-6	151	744	344	0.47	0.74	0.58
5	0.014	0.015 (2)	0.014	9e-8	1e-5	5e-6	244	2436	1176	0.55	1.09	0.82
6	0.013	0.014	0.014	2e-7	2e-5	7e-6	361	2135	1184	0.62	1.05	0.84
7	0.012	0.014	0.014	1e-8	2e-5	7e-6	321	3119	1368	0.61	1.19	0.89

methods II and IV made it worse compared to the unregularized approach.

Generalization: All regularization approaches improve the generalization compared to the unregularized problem by preventing oscillation in the final mapping generated by the MLP. When n_1 is increased, regularization methods I and III seem to be more stable and thus more robust than methods II and IV.

Conclusion: From the four regularization methods tested, I and III are preferable to II and IV in every respect. One cannot make a difference between I and III, and we note that centering the input data already decreases the significance of the hidden bias.

4.2 Classification.

Example 2. We consider the well-known benchmark to classify Iris flowers (cf. Gupta & Lam, 1998) according to measurements obtained from the UCI repository (Blake & Merz, 1998). In this example, we have $n_2 = 3$ (number of classes), $n_0 = 4$ (number of features), and initially 50 samples from each of the three classes. Due to the choice of the k-tanh activation functions prescaling of the output data $\{y_i\}$ into the range $[-1, 1]$ destroys the linear independence between different classes.

In Tables 6 through 11, we have solved the classification (optimization) problem again 10 times using 40 random samples from each class as the learning set and the remaining 10 samples from each class as the test set. These two data sets have been formed using five different random permutations of the initial data realizing a simple cross-validation technique. Hence, we have $N = 120$ and the remaining 30 samples in the test set for each permutation.

In Tables 9 through 11, we have added noise to the inputs in permuted learning sets. First, means with regard to each four features within the three classes have been computed. After that, the class mean vectors have been multiplied component-wise with a noise vector $\delta\,\varepsilon_i$ for $\delta = 0.3$ and $\varepsilon_i \in \mathcal{N}(0, 1)$, and this has been added to the unscaled learning data.

To this end, we derive some final observations based on the computational results, especially in example 2:

Local minima: In all numerical tests for unregularized and regularized learning problems, the L-BFGS optimization algorithm was convergent. This suggests that by using the proposed combination of linear final layer, prescaling, and mixed activation, we are able to deal with the flat error surfaces during the training. Furthermore, because usually in all 10 test runs of one learning problem, a different value of the cost functional is obtained, the different results are revealing "true"

Table 6: Computational Results in Example 2 Without Regularization for $n_1 = 9$.

P	C_L			C		Its			CPU		
	Minimum	Maximum	Mean	C_L	C_T	Minimum	Maximum	Mean	Minimum	Maximum	Mean
1	0	0	0	0	2	109	534	239	0.37	0.60	0.47
2	0	0	0	0	3	81	664	290	0.30	0.65	0.49
3	0	1	1	0	1	106	1243	492	0.35	0.93	0.57
4	0	0	0	0	2	129	891	349	0.36	0.75	0.51
5	0	1	1	0	0	92	702	286	0.31	0.67	0.48
Mean	0	0.4	0.4	0	1.6	103	807	331	0.34	0.72	0.50

Notes: The first column, P, contains the permutation index. In the second column, C_L, minimum, maximum, and (rounded) mean values for the number of false classifications in the learning set are given. The third column, C, includes numbers of false classifications in the learning and test sets (denoted with C_L and C_T) for the optimal perceptron $\mathcal{N}_P^*$ satisfying $C_L(\mathcal{N}_P^*)+C_T(\mathcal{N}_P^*) \leq C_L(\mathcal{N}_P) + C_T(\mathcal{N}_P)$ over all networks encountered during the 10 optimization runs for the current permutation P.

Table 7: Computational Results in Example 2 for Regularization I with $\varepsilon = 10^{-3}$ and $n_1 = 9$.

P	C_L			C		Its			CPU		
	Minimum	Maximum	Mean	C_L	C_T	Minimum	Maximum	Mean	Minimum	Maximum	Mean
1	0	0	0	0	2	517	1280	843	0.60	0.93	0.76
2	0	0	0	0	3	443	814	551	0.57	0.72	0.63
3	0	0	0	0	0	742	1797	1047	0.68	1.04	0.82
4	0	0	0	0	2	426	1451	792	0.56	0.91	0.71
5	0	1	0	0	0	646	1604	884	0.69	1.00	0.76
Mean	0	0.2	0	0	1.4	555	1389	823	0.62	0.92	0.74

Table 8: Computational Results in Example 2 for Regularization III with $\varepsilon = 10^{-3}$ and $n_1 = 9$.

P	C_L			C		Its			CPU		
	Minimum	Maximum	Mean	C_L	C_T	Minimum	Maximum	Mean	Minimum	Maximum	Mean
1	0	0	0	0	2	369	939	543	0.56	0.77	0.61
2	0	0	0	0	3	484	923	602	0.58	0.80	0.64
3	0	2	1	0	0	418	868	677	0.56	0.78	0.67
4	0	0	0	0	3	382	643	507	0.54	0.69	0.60
5	0	3	1	0	0	413	973	619	0.56	0.82	0.64
mean	0	1	0.4	0	1.6	413	869	590	0.56	0.77	0.63

Table 9: Computational Results in Example 2 Without Regularization for $n_1 = 9$ and Noisy Learning Set.

	C_L			C		Its			CPU		
P	Minimum	Maximum	Mean	C_L	C_T	Minimum	Maximum	Mean	Minimum	Maximum	Mean
1	0	1	0	0	2	275	3008	968	0.49	1.19	0.73
2	0	1	1	0	2	240	460	339	0.47	0.57	0.52
3	0	3	1	0	1	175	2680	948	0.41	1.16	0.73
4	0	1	1	0	2	146	1655	589	0.39	0.98	0.61
5	0	2	1	0	0	284	5464	927	0.50	1.42	0.65
Mean	0	1.6	0.8	0	1.4	224	2653	754	0.45	1.06	0.65

Table 10: Computational Results in Example 2 for Regularization I with $\varepsilon = 10^{-3}$, $n_1 = 9$, and Noisy Learning Set.

	C_L			C		Its			CPU		
P	Minimum	Maximum	Mean	C_L	C_T	Minimum	Maximum	Mean	Minimum	Maximum	Mean
1	0	1	0	0	2	492	1196	724	0.59	0.88	0.68
2	0	0	0	0	3	486	1059	782	0.58	0.81	0.70
3	0	0	0	0	1	579	982	739	0.62	0.78	0.69
4	0	1	0	0	2	616	1120	862	0.64	0.84	0.74
5	0	1	1	0	0	464	1211	756	0.58	0.88	0.69
Mean	0	0.6	0.2	0	1.6	527	1114	773	0.60	0.84	0.70

Table 11: Computational Results in Example 2 for Regularization III with $\varepsilon = 10^{-3}$, $n_1 = 9$, and Noisy Learning Set.

	C_L			C		Its			CPU		
P	Minimum	Maximum	Mean	C_L	C_T	Minimum	Maximum	Mean	Minimum	Maximum	Mean
1	0	1	0	0	2	434	1024	710	0.56	0.80	0.68
2	0	2	1	0	3	365	941	580	0.54	0.76	0.62
3	0	3	1	0	1	464	800	653	0.58	0.71	0.65
4	0	2	1	0	2	386	1007	725	0.54	0.83	0.69
5	0	1	1	0	0	458	1003	690	0.58	0.79	0.67
mean	0	1.8	0.8	0	1.6	421	955	672	0.56	0.78	0.66

local minima instead of just some symmetrical representations of the same MLP transformation.

Efficiency: By comparing the number of iterations and the CPU time, the unregularized problem seems to be easier to solve than the regularized one (with given β) for the clean learning data. On average, there is no real difference between the unregularized and regularized problems for the noisy learning data, but the variation in the number of iterations is significantly larger for the unregularized approach.

Comparison: There seems to be no significant difference between the quality of perceptrons resulting from the unregularized and regularized problems, with or without additional noise. This suggests that the iris learning data are quite stable and contain only a few cases with nongaussian degradations. Moreover, according to the observations in example 1, one reason for the quite similar behavior can be the size of n_1, which is not very large compared to n_0 and n_2. Finally, we cannot favor either of the two regularization methods I and III according to these results.

Generalization: The amount of false classifications in the test set is between 0 and 3, that is, between 0% and 10% in all test runs. A single preferable choice (an ensemble, of course, is another possibility) for an MLP classifier would probably be the one with median of false classifications in C_T obtained by using permutation 1 or 4, even if for the fifth permutation, an optimal classifier was found according to the given data. Notice that the amount of variation in C_T over the different permutations provides useful information on the quality of the data.

Appendix A: Proofs of Lemmas

Proof of Lemma 1. Functional $J(\mathbf{W})$ in a component-wise form reads as

$$J(\mathbf{W}) = \frac{1}{2}\sum_{i=1}^{m_2}\left(\sum_{j=1}^{m_1} w_{ij}\,v_j - y_i\right)^2,$$

where i represent the row and j the column index of $\mathbf{W}$, respectively. A straightforward calculation shows that

$$\frac{\partial J}{\partial w_{ik}} = \left(\sum_{j=1}^{m_1} w_{ij}\,v_j - y_i\right) v_k \Rightarrow \frac{\partial J}{\partial \mathbf{w}_{i,:}} = \left(\sum_{j=1}^{m_1} w_{ij}\,v_j - y_i\right)\mathbf{v}^T$$
$$= [\mathbf{W}\,\mathbf{v} - \mathbf{y}]_i\,\mathbf{v}^T \Rightarrow \frac{\partial J}{\partial \mathbf{W}} = [\mathbf{W}\,\mathbf{v} - \mathbf{y}]\,\mathbf{v}^T. \tag{A.1}$$

Here we have used Matlab-type abbreviations for the ith row vector $\mathbf{w}_{i,:}$ of matrix $\mathbf{W}$.

Proof of Lemma 2. As above, consider the component-wise form of the functional

$$J(\mathbf{u}) = \frac{1}{2}\sum_{j=1}^{m_2}\left(\sum_{i=1}^{m_1} w_{ji}\,f_i(u_i) - y_j\right)^2 = \sum_{j=1}^{m_2} J_j(\mathbf{u}), \tag{A.2}$$

where $J_j(\mathbf{u}) = \frac{1}{2}\sum_{i=1}^{m_1}(w_{ji}\,f_i(u_i) - y_j)^2 = \frac{1}{2}[\mathbf{W}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}]_j^2$. Then

$$\frac{\partial J_j}{\partial u_k} = \left[\sum_{i=1}^{m_1} w_{ji}\,f_i(u_i) - y_j\right] w_{jk}\,f_k^{'}(u_k) = w_{jk}\,f_k^{'}(u_k)\,[\mathbf{W}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}]_j. \tag{A.3}$$

Remember here that $\mathbf{u}$ is a vector with m_2 components. As in lemma 1, we get the derivative of $J(\mathbf{u})$ with respect to $\mathbf{u}$ by treating k as row index and j as column index. Proceeding like this, we obtain from equation A.3

$$\nabla_{\mathbf{u}} J(\mathbf{u}) = \begin{pmatrix} w_{11}\,f_1^{'}(u_1) & \dots & w_{m_2 1}\,f_1^{'}(u_1) \\ \vdots & \ddots & \vdots \\ w_{1m_1}\,f_{m_1}^{'}(u_{m_1}) & \dots & w_{m_2 m_1}\,f_{m_1}^{'}(u_{m_1}) \end{pmatrix} [\mathbf{W}\mathcal{F}(\mathbf{u}) - \mathbf{y}]$$
$$= \left(\mathbf{W}\,\mathcal{F}^{'}(\mathbf{u})\right)^T [\mathbf{W}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}], \tag{A.4}$$

which is the desired result when $(\mathbf{W}\,\mathcal{F}^{'}(\mathbf{u}))^T$ is replaced for $\mathrm{Diag}\{\mathcal{F}^{'}(\mathbf{u})\}\,\mathbf{W}^T$.

Proof of Lemma 3. Here we introduce the basic Lagrangian technique to simplify the calculations. As the first step, we define the extra variable $\mathbf{u} = \mathbf{W}\mathbf{v}$ and instead of the original problem consider the equivalent constraint optimization problem,

$$\min_{(\mathbf{u},\mathbf{W})} \tilde{J}(\mathbf{u}, \mathbf{W}) = \frac{1}{2}\,\|\bar{\mathbf{W}}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}\|^2 \quad \text{subject to} \;\; \mathbf{u} = \mathbf{W}\,\mathbf{v}, \tag{A.5}$$

where $\mathbf{u}$ and $\mathbf{W}$ are treated as independent variables linked together by the given constraint. The Lagrange functional associated with equation A.5 reads as

$$\mathcal{L}(\mathbf{u}, \mathbf{W}, \boldsymbol{\lambda}) = \tilde{J}(\mathbf{u}, \mathbf{W}) + \boldsymbol{\lambda}^T(\mathbf{u} - \mathbf{W}\,\mathbf{v}), \tag{A.6}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^{m_1}$ contains the Lagrangian variables for the constraint.

Noticing that the values of functions $\tilde{J}(\mathbf{u}, \mathbf{W})$ in equation A.5 and $\mathcal{L}(\mathbf{u}, \mathbf{W}, \boldsymbol{\lambda})$ in equation A.6 coincide if the constraint $\mathbf{u} = \mathbf{W}\,\mathbf{v}$ is satisfied, it follows that a solution of equation A.5 is equivalently characterized by the saddle-point conditions $\nabla\mathcal{L}(\mathbf{u}, \mathbf{W}^1, \boldsymbol{\lambda}) = 0$. Therefore, we compute the derivatives $\nabla_{\mathbf{u}}\mathcal{L}$, $\nabla_{\mathbf{W}}\mathcal{L}$, and $\nabla_{\boldsymbol{\lambda}}\mathcal{L}$ (in a suitable form) for the Lagrangian.

The gradient vector $\nabla_{\mathbf{u}}\mathcal{L}(\mathbf{u}, \mathbf{W}, \boldsymbol{\lambda})$ is, due to lemma 2, of the form

$$\nabla_{\mathbf{u}}\mathcal{L} = \nabla_{\mathbf{u}}\tilde{J}(\mathbf{u}) + \boldsymbol{\lambda} = \mathrm{Diag}\{\mathcal{F}'(\mathbf{u})\}\,\bar{\mathbf{W}}^T\,[\mathbf{W}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}] + \boldsymbol{\lambda}. \tag{A.7}$$

Other derivates for the Lagrangian are given by $\nabla_{\mathbf{W}}\mathcal{L} = -\boldsymbol{\lambda}\,\mathbf{v}^T$ and $\nabla_{\boldsymbol{\lambda}}\mathcal{L} = \mathbf{u} - \mathbf{W}\mathbf{v}$. Using formula $-\boldsymbol{\lambda} = \mathrm{Diag}\{\mathcal{F}'(\mathbf{u})\}\,\bar{\mathbf{W}}^T\,[\mathbf{W}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}]$ due to equation A.7 and substituting this into $\nabla_{\mathbf{W}}\mathcal{L}$, we obtain

$$\nabla_{\mathbf{W}}\mathcal{L} = \mathrm{Diag}\{\mathcal{F}'(\mathbf{u})\}\,\bar{\mathbf{W}}^T\,[\bar{\mathbf{W}}\,\mathcal{F}(\mathbf{u}) - \mathbf{y}]\,\mathbf{v}^T. \tag{A.8}$$

Due to the equivalency of equations A.5 and A.6 with the original problem, the desired gradient matrix is given by equation A.8 when $\mathbf{W}\mathbf{v}$ is substituted for $\mathbf{u}$.

Proof of Lemma 4. To simplify the calculations, we now introduce two extra variables $\mathbf{u} = \mathbf{W}\mathbf{v}$ and $\tilde{\mathbf{u}} = \tilde{\mathbf{W}}\,\tilde{\mathcal{F}}(\mathbf{u}) = \tilde{\mathbf{W}}\,\tilde{\mathcal{F}}(\mathbf{W}\mathbf{v})$. As in the previous proof, we first consider the constraint optimization problem:

$$\min_{(\mathbf{u},\tilde{\mathbf{u}},\mathbf{W})} \tilde{J}(\mathbf{u}, \tilde{\mathbf{u}}, \mathbf{W}) = \frac{1}{2}\,\|\bar{\mathbf{W}}\,\bar{\mathcal{F}}(\tilde{\mathbf{u}}) - \mathbf{y}\|^2 \quad \text{s.t. } \mathbf{u} = \mathbf{W}\,\mathbf{v},\ \tilde{\mathbf{u}} = \tilde{\mathbf{W}}\,\tilde{\mathcal{F}}(\mathbf{u}). \tag{A.9}$$

The Lagrange functional associated with equation A.9 reads as

$$\mathcal{L}(\mathbf{u}, \tilde{\mathbf{u}}, \mathbf{W}, \boldsymbol{\lambda}, \tilde{\boldsymbol{\lambda}}) = \tilde{J}(\mathbf{u}, \tilde{\mathbf{u}}, \mathbf{W}) + \boldsymbol{\lambda}^T(\mathbf{u} - \mathbf{W}\,\mathbf{v}) + \tilde{\boldsymbol{\lambda}}^T(\tilde{\mathbf{u}} - \tilde{\mathbf{W}}\,\tilde{\mathcal{F}}(\mathbf{u})). \tag{A.10}$$

Using similar techniques as in the previous proof, derivates for the saddle-point conditions of the Lagrangian are given by $\nabla_{\mathbf{u}}\mathcal{L} = -[\tilde{\mathbf{W}}\,\tilde{\mathcal{F}}'(\mathbf{u})]^T\,\tilde{\boldsymbol{\lambda}} + \boldsymbol{\lambda}$, $\nabla_{\tilde{\mathbf{u}}}\mathcal{L} = [\bar{\mathbf{W}}\,\bar{\mathcal{F}}'(\tilde{\mathbf{u}})]^T\,[\bar{\mathbf{W}}\,\bar{\mathcal{F}}(\tilde{\mathbf{u}}) - \mathbf{y}] + \tilde{\boldsymbol{\lambda}}$, $\nabla_{\mathbf{W}}\mathcal{L} = -\boldsymbol{\lambda}\,\mathbf{v}^T$, $\nabla_{\boldsymbol{\lambda}}\mathcal{L} = \mathbf{u} - \mathbf{W}\mathbf{v}$, and $\nabla_{\tilde{\boldsymbol{\lambda}}}\mathcal{L} = \tilde{\mathbf{u}} - \tilde{\mathbf{W}}\,\tilde{\mathcal{F}}(\mathbf{u})$. From $\nabla_{(\mathbf{u},\tilde{\mathbf{u}})}\mathcal{L} = \mathbf{O}$ we obtain

$$\boldsymbol{\lambda} = [\tilde{\mathbf{W}}\,\tilde{\mathcal{F}}'(\mathbf{u})]^T\,\tilde{\boldsymbol{\lambda}} = -[\tilde{\mathbf{W}}\,\tilde{\mathcal{F}}'(\mathbf{u})]^T\,[\bar{\mathbf{W}}\,\bar{\mathcal{F}}'(\tilde{\mathbf{u}})]^T\,[\bar{\mathbf{W}}\,\bar{\mathcal{F}}(\tilde{\mathbf{u}}) - \mathbf{y}], \tag{A.11}$$

which, substituted into $\nabla_{\mathbf{W}}\mathcal{L}$, yields

$$\nabla_{\mathbf{W}}\mathcal{L} = \mathrm{Diag}\{\tilde{\mathcal{F}}'(\mathbf{u})\}\,\tilde{\mathbf{W}}^T\,\mathrm{Diag}\{\bar{\mathcal{F}}'(\tilde{\mathbf{u}})\}\,\bar{\mathbf{W}}^T\,[\bar{\mathbf{W}}\,\bar{\mathcal{F}}(\tilde{\mathbf{u}}) - \mathbf{y}]\,\mathbf{v}^T. \tag{A.12}$$

This, together with the original expressions $\mathbf{u} = \mathbf{W}\,\mathbf{v}$ and $\tilde{\mathbf{u}} = \tilde{\mathbf{W}}\,\tilde{\mathcal{F}}(\mathbf{u})$, proves the result.

Acknowledgments

I thank Pasi Koikkalainen for many enlightening discussions concerning ANNs. The comments and suggestions of the unknown referee improved and clarified the presentation significantly, especially through the excellent reference Orr and Müller (1998).

References

Bazaraa, M. S., Sherali, H. D., & Shetty, C. M. (1993). *Nonlinear programming: Theory and algorithms* (2nd ed.). New York: Wiley.
Bertsekas, D. P. (1982). *Constrained optimization and Lagrange multiplier methods*. New York: Academic Press.
Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford: Clarendon Press.
Blake, C. L., & Merz, C. J. (1998). *UCI repository of machine learning databases*. Available on-line: http://www.ics.uci.edu/~mlearn/MLRepository.html.
Byrd, R. H., Lu, P., & Nocedal, J. (1995). A limited memory algorithm for bound constrained optimization, *SIAM Journal on Scientific and Statistical Computing*, *5*, 1190–1208.
Caruna, R. (1998). A dozen tricks with multitask learning. In G. B. Orr & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade*. Berlin: Springer-Verlag.
Cataltepe, Z., Abu-Mostafa, Y. S., & Magdon-Ismail, M. (1999). No free lunch for early stopping. *Neural Computation*, *11*, 995–1009.
Cavalieri, S., & Mirabella, O. (1999). A novel learning algorithm which improves the partial fault tolerance of multilayer neural networks. *Neural Networks*, *12*, 91–106.
Chambolle, A., & Lions, P. L. (1997). Image recovery via total variation minimization and related problems. *Numerische Mathematik*, *76*, 167–188.
Chen, D. S., & Jain, R. C. (1994). A robust back propagation learning algorithm for function approximation. *IEEE Transactions on Neural Networks*, *5*, 467–479.
de Villiers, J., & Barnard, E. (1992). Backpropagation neural networks with one and two hidden layers. *IEEE Transactions on Neural Networks*, *1*, 136–141.
Diamantaras, K. I., & Kung, S. Y. (1996). *Principal component neural networks: Theory and applications*. New York: Wiley.
Di Martino, M., Fanelli, S., & Protasi, M. (1996). Exploring and comparing the best "direct methods" for the efficient training of MLP-networks. *IEEE Transactions on Neural Networks*, *7*, 1497–1502.
Flake, G. W. (1998). Square unit augmented, radially extended, multilayer perceptrons. In G. B. Orr & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade*. Berlin: Springer-Verlag.
Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, *4*, 1–58.
Gilbarg, D., & Trudinger, N. S. (1983). *Elliptic partial differential equations of second order*. Berlin: Springer-Verlag.
Gorse, D., Shepherd, A. J., & Taylor, J. G. (1997). The new ERA in supervised learning. *Neural Networks*, *10*, 343–352.
Goutte, C., & Hansen, L. K. (1997). Regularization with a pruning prior. *Neural Networks*, *10*, 1053–1059.
Griewank, A. (2000). *Evaluating derivatives: Principles and techniques of algorithmic differentiation*. Philadelphia: SIAM.
Gupta, A., & Lam, S. M. (1998). Weight decay backpropagation for noisy data. *Neural Networks*, *11*, 1527–1137.

Hagan, M. T., & Menhaj, M. B. (1994). Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks, 5*, 989–993.

Haykin, S. (1994). *Neural networks; A comprehensive foundation*. New York: Macmillan.

Hintz-Madsen, M., Hansen, L. K., Larsen, J., Pedersen, M. W., & Larsen, M. (1998). Neural classifier construction using regularization, pruning and test error estimation. *Neural Networks, 11*, 1659–1670.

Holmström, L., Koistinen, P., Laaksonen, J., & Oja, E. (1997). Neural and statistical classifiers—taxonomy and two case studies. *IEEE Transactions on Neural Networks, 8*, 5–17.

Horn, D., Naftaly, U., & Intrator, N. (1998). Large ensemble averaging. In G. B. Orr & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade*. Berlin: Springer-Verlag.

Japkowicz, N., Hanson, S. J., & Gluck, M. A. (2000). Nonlinear autoassociation is not equivalent to PCA. *Neural Computation, 12*, 531–545.

Kärkkäinen, T. (2000). *MLP-network in a layer-wise form: Derivations, consequences, and applications to weight decay* (Tech. Rep. No. C1). Jyväskylä, Finland: Department of Mathematical Information Technology, University of Jyväskylä.

Kärkkäinen, T., & Majava, K. (2000a). Nonmonotone and monotone active-set methods for image restoration, Part 1: Convergence analysis. *Journal of Optimization Theory and Applications, 106*, 61–80.

Kärkkäinen, T., & Majava, K. (2000b). Nonmonotone and monotone active-set methods for image restoration, Part 2: Numerical results. *Journal of Optimization Theory and Applications, 106*, 81–105.

Kärkkäinen, T., Majava, K., & Mäkelä, M. M. (2001). Comparison of formulations and solution methods for image restoration problems. *Inverse Problems, 17*, 1977–1995.

Kärkkäinen, T., & Toivanen, J. (2001). Building blocks for odd-even multigrid with applications to reduced systems. *Journal of Computational and Applied Mathematics, 131*, 15–33.

Kosko, B. (1992). *Neural networks and fuzzy systems: A dynamical systems approach to machine intelligence*. Englewood Cliffs, NJ: Prentice-Hall.

Kwon, T. M., & Cheng, H. (1996). Contrast enhancement for backpropagation. *IEEE Transactions on Neural Networks, 7*, 515–524.

LeCun, Y., Bottous, L., Orr, G. B., & Müller, K.-R. (1998). Efficient backprop. In G. B. Orr & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade*. Berlin: Springer-Verlag.

Liano, K. (1996). Robust error measure for supervised neural network learning with outliers. *IEEE Transactions on Neural Networks, 7*, 246–250.

Liu, Y., & Yao, X. (1999). Ensemble learning via negative correlation. *Neural Networks, 12*, 1399–1404.

MacKay, D. J. C. (1992). A practical Bayesian framework for backpropagation networks. *Neural Computation, 4*, 448–472.

Magoulas, G. D., Vrahatis, M. N., & Androulakis, G. S. (1999). Improving the convergence of the backpropagation algorithm using learning rate adaptation methods. *Neural Computation, 11*, 1769–1796.

Mäkelä, M. M., & Neittaanmäki, P. (1992). *Nonsmooth optimization: Analysis and algorithms with applications to optimal control*. Singapore: World Scientific.

McKeown, J. J., Stella, F., & Hall, G. (1997). Some numerical aspects of the training problem for feed-forward neural nets. *Neural Networks, 10*, 1455–1463.

McLean, D., Bandar, Z., & O'Shea, J. D. (1998). An empirical comparison of back propagation and the RDSE algorithm on continuously valued real world data. *Neural Networks, 11*, 1685–1694.

Moody, J. O., & Antsaklis, P. J. (1996). The dependence identification neural network construction algorithm. *IEEE Transactions on Neural Networks, 7*, 3–15.

Neal, R. M. (1996). *Bayesian learning for neural networks*. New York: Springer.

Nocedal, J., & Wright, S. J. (1999). *Numerical optimization*. New York: Springer.

Ormoneit, D. (1999). A regularization approach to continuous learning with an application to financial derivatives pricing. *Neural Networks, 12*, 1405–1412.

Orr, G. B., & Müller, K.-R. (Eds.). (1998). *Neural networks: Tricks of trade*. Berlin: Springer-Verlag.

Osman, I. H., & Kelly, J. P. (Eds.) (1996). *Meta-heuristics: Theory and applications*. Norwell, MA: Kluwer.

Park, D. C., El-Sharkawi, M. A., & Marks II, R. J. (1991). An adaptively trained neural network. *IEEE Transactions on Neural Networks, 2*, 334–345.

Prechelt, L. (1998). Early stopping. In G. B. Orr & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade*. Berlin: Springer-Verlag.

Reed, R. D., & Marks, II, R. J. (1999). *Neural smithing: Supervised learning in feedforward artificial neural networks*. Cambridge, MA: MIT Press.

Rögnvaldsson, T. S. (1998). A simple trick for estimating the weight decay parameter. In G. B. Orr & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade*. Berlin: Springer-Verlag.

Rohas, R. (1996). *Neural networks: A systematic introduction*. Berlin: Springer-Verlag.

Saito, K., & Nakano, R. (2000). Second-order learning algorithm with squared penalty term. *Neural Computation, 12*, 709–729.

Schaal, S., & Atkeson, C. G. (1998). Constructive incremental learning from only local information. *Neural Computation, 10*, 2047–2084.

Scherzer, O., Engl, H. W., & Kunisch, K. (1993). Optimal a posteriori parameter choice for Tikhonov regularization for solving nonlinear ill-posed problems. *SIAM Journal on Numerical Analysis, 30*, 1796–1838.

Tamura, S., & Tateishi, M. (1997). Capabilities of a four-layered feedforward neural network: Four layers versus three. *IEEE Transactions on Neural Networks, 8*, 251–255.

Tetko, I. V., & Villa, A. E. P. (1997). Efficient partition of learning data sets for neural network training. *Neural Networks, 10*, 1361–1374.

Thimm, G., & Fiesler, E. (1997). High-order and multilayer perceptron initialization. *IEEE Transactions on Neural Networks, 8*, 349–359.

van de Laar, P., & Heskes, T. (1999). Pruning using parameter and neuronal metrics. *Neural Computation, 11*, 977–993.

van der Smagt, P., & Hirzinger, G. (1998). Solving the ill-conditioning in neural network learning. In G. B. Orr & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade*. Berlin: Springer-Verlag.

Vitela, J. E., & Reifman, J. (1997). Premature saturation in backpropagation networks: Mechanism and necessary conditions. *Neural Networks, 10*, 721–735.

Wang, G.-J., & Chen, C.-C. (1996). A fast multilayer neural-network training algorithm based on the layer-by-layer optimizing procedures. *IEEE Transactions on Neural Networks, 7*, 768–775.

Wang, Y.-J., & Lin, C.-T. (1998). A second-order learning algorithm for multilayer networks based on block Hessian matrix. *Neural Networks, 11*, 1607–1622.

Yaeger, L. S., Webb, B. J., & Lyon, R. F. (1998). Combining neural networks and context-driven search for on-line printed handwriting recognition in the Newton. In G. B. Orr & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade*. Berlin: Springer-Verlag.

Zhang, J., & Morris, A. J. (1998). A sequential learning approach for single hidden layer neural networks. *Neural Networks, 11*, 65–80.

Received November 20, 2000; accepted October 12, 2001.