Int. J. Appl. Math. Comput. Sci., 2000, Vol.10, No.4, 779–790

ADAPTATION OF REGULARIZATION PARAMETERS IN NM-DELTA NETWORKS

PIOTR GOŁĄBEK*,**, WITOLD KOSIŃSKI**,***

The paper describes an application of regularization techniques to an automatic choice of parameters driving the learning process in the NM-Delta neural network architecture. The heterogeneous learning algorithm is identified as very similar to the Levenberg-Marquardt method but with a considerably smaller computational cost and different justification of parameter selection. The performance of the modified algorithm proves to be comparable with that of the Levenberg-Marquardt.

Keywords: NM-Delta, M-Delta, regularization, Levenberg-Marquardt, quasi-Newton

1. Introduction

In this paper we apply some results from regularization theory to improve the learning process taking place in a specific neural network structure. The regularization approach proves to be relatively successful when applied to the interpretation of the learning processes. Many techniques used in neural networks, such as weight decay, weight pruning, learning with noise, radial basis function models, etc., have been explained in terms of the regularization. In fact, regularization theory and the Bayesian inference constitute a very promising and coherent framework for the analysis of neural networks.

In this paper, we analyze the optimization process in NM-Delta neural networks. The learning algorithm used by such a network can be considered in the perspective drawn from different fields of research: adaptive signal processing, optimization and regularization theories. Our goal is to cross-examine this different approaches in order to build an effective learning process. In particular, we apply regularization techniques to enrich the original algorithm with adaptation of the regularization parameter.

* Radom Technical University, Division of Electronics and Automatics, ul. Malczewskiego 29, 26–600 Radom, Poland, e-mail: pgolab@ippt.gov.pl

** Polish Academy of Sciences, Division of Vision and Measurement Systems, Institute of Fundamental Technological Research, ul. Świętokrzyska 21, 00–049 Warsaw, Poland

*** Polish-Japanese Institute of Information Technology, ul. Koszykowa 86, 02–008 Warsaw, Poland, e-mail: wkos@pjwstk.waw.pl

2. Description of the NM-Delta Algorithm and Notation

2.1. M-Delta Algorithm

The M-Delta neural network (Kosiński and Weigl, 1996; 1998; Weigl, 1995) is a network with one hidden layer and neurons employing the generalized logistic activation function

$$f(s) = \frac{m}{1 + e^{-\delta s}}. \tag{1}$$

The additional parameters m and δ determining the slope and the range of the activation functions are adapted along with the regular weights using the Gradient Descent Method, i.e. updating m and δ in the p-th iteration of the algorithm according to the general formula

$$\begin{aligned} \delta_i(p) &= \delta_i(p-1) - \mu_{\delta_i} \frac{\partial F}{\partial \delta_i}(p-1), \\ m_i(p) &= m_i(p-1) - \mu_{m_i} \frac{\partial F}{\partial m_i}(p-1), \end{aligned} \tag{2}$$

where F is the error function and μ is the learning speed factor. The M-Delta network is therefore an extension of the ordinary backpropagation (BP) network. Updating the m and δ parameters usually gives a boost to the learning process because, as has been shown in various publications (Gołąbek *et al.*, 1999; Krushke and Movellan, 1991; Thimm *et al.*, 1996), changing these parameters can be treated as an equivalent of the adaptation of the learning speed for weights in regular BP networks. An additional practical advantage of this architecture is the elimination of the necessity of data scaling. An M-Delta network generally uses one hidden layer, as it is sufficient for building a universal approximator for a broad class of functions, and the function approximation is the main application area for this architecture.

2.2. NM-Delta Algorithm

The NM-Delta network (Kosiński and Weigl, 1996; Weigl 1995) adds complexity to the M-Delta approach by using a heterogeneous learning algorithm consisting of two phases:

- adaptation of the regular weights by means of the least-squares algorithm (the m and δ parameters are left intact during this phase),
- adaptation of the m and δ parameters (with frozen regular weights) by means of the gradient descent method (GDM), in exactly the same way as in M-Delta networks.

The use of the least-squares algorithm to calculate optimal weight values comes from partial linearization of the learning problem, as it will be discussed in detail in Section 3. Strong assumptions implicitly cast on the learning problem during the least-squares phase are then relaxed by the second phase (m and δ adaptation). The least-squares algorithm used in NM-Delta networks is implemented as a recursive procedure

(RLS), calculating the pseudoinverse of the autocorrelation matrix by means of the well-known matrix inversion lemma (Haykin, 1996; Rutkowski, 1994; Weigl, 1995). This lemma allows for iterative construction of the matrix inverse, without actually performing any inversion operation, which greatly decreases the computational cost of the algorithm.

Owing to the RLS phase, NM-Delta networks are characterized by very good convergence speeds. The drawback of both the architectures (M-Delta and NM-Delta) is that many extra network parameters are introduced. Dealing with these parameters is troublesome, as they have to be properly managed in terms of the initial value assignment and driving the adaptation process. Our paper proposes a solution to this problem.

2.3. Notation

The original NM-Delta algorithm uses the stochastic adaptation scheme, making updates after presentation of each training pair. The contribution from subsequent pairs is exponentially weighted by the so-called forgetting factor. The motivation for this parameter comes from adaptive signal processing, where the RLS algorithm is used extensively. The parameter is used when dealing with nonstationary signals. In this paper, we do not attempt to consider stochastic aspects of the NM-Delta algorithm and thus will use a modification of the original algorithm with a batch (deterministic) update scheme. The forgetting factor will be held equal to one, which means no forgetting.

Whenever there is a reference to the Hessian, this usually means some more or less crude approximation to the true Hessian matrix (the matrix of the second derivatives of the error function w.r.t. the weights). We usually consider a block-diagonal approximation to the Hessian with zero off-diagonal block elements.

The paper uses the following notation:

x_i – the i-th input to the network,

w_{ij} – the weight of the connection from the j-th to the i-th neuron,

z_i – the net activation of the i-th neuron,

u_i – the output of the i-th neuron,

F – the error function,

$\gamma_i = -\partial F/\partial z_i$ – the error of the linear part of the i-th neuron, and

$g_i = -\partial F/\partial u_i$ – the error of the nonlinear part of the i-th neuron.

For notational convenience, the superscript denoting the layer number is omitted (with one exception) in all formulae. For the same reason, the formal dependence of various quantities on the index p of the learning epoch is pointed out only if necessary.

The reader is warned that some original notation found in cited works had to be changed in this article in order to avoid notational conflicts.

3. RLS Phase of the NM-Delta Algorithm

The RLS phase of the NM-Delta algorithm tries to calculate the optimal weights for the processed layer of neurons by solving the system of equations $\partial F/\partial w_{ij} = 0$. As will be shown, the RLS phase effectively implements a quasi-Newton procedure which converges toward the optimal weights iteratively calculating in each iteration consecutive approximations by

$$\boldsymbol{w}(p+1) = \boldsymbol{w}(p) - \boldsymbol{H}^{-1}(p)\boldsymbol{J}(p), \tag{3}$$

where $\boldsymbol{H}(p)$ and $\boldsymbol{J}(p)$ are respectively the Hessian and the Jacobian of the error function w.r.t. the weights in the p-th learning epoch. Using such a formula, the Newton algorithm assumes that the error function is quadratic in the weights. The RLS phase is built on another approach. It replaces the gradient zeroing condition by

$$\frac{\partial F}{\partial w_{ik}} = \frac{\partial F}{\partial z_i}\frac{\partial z_i}{\partial w_{ik}} = (\eta_i - z_i)\frac{\partial z_i}{\partial w_{ik}}\,. \tag{4}$$

Here η_i is the desired value of the net-activation for the i-th neuron calculated by means of the back-propagation or the antithreshold function. It is clear that the algorithm makes quite strong assumptions on the error function. It assumes that this function is quadratic in z_i and that it has the second derivative $\partial^2 F/\partial z_i^2 = 1$. Moreover, the RLS algorithm operates separately on each neuron, thus assuming implicitly the diagonal character of the $[\partial^2 F/\partial z_i^2]$ matrix. These assumptions can be relaxed by an appropriate design of the m and δ adaptation phase, as will be shown in what follows.

Accepting the assumptions, we can interpret the RLS algorithm as a quasi-Newton procedure in which there is a gradually improving Hessian estimation. Starting with the assumption $\partial^2 F/\partial z_i \partial z_j = \boldsymbol{I}$, we can write

$$\frac{\partial^2 F}{\partial w_{ij}\partial w_{ik}} = x_i x_k \frac{\partial^2 F}{\partial z_j \partial z_i} = x_i x_k \delta_i^j, \tag{5}$$

where δ_i^j is the Kronecker delta. We then get a block-diagonal form of the estimate to the Hessian, with the partial autocorrelation matrix $\boldsymbol{A}(p) = \boldsymbol{X}(p)\boldsymbol{X}^T(p)$ repeated for each neuron. The solution to the optimal weight estimates for each neuron takes the form

$$\hat{\boldsymbol{w}}_j(p) = \hat{\boldsymbol{w}}_j(p-1) - \boldsymbol{S}(p)\boldsymbol{J}_j(p), \tag{6}$$

where the matrix $\boldsymbol{S}(p)$ is a gradually improving estimate of $\boldsymbol{A}^{-1}(p)$, and $\boldsymbol{J}_j(p)$ is the gradient vector

$$\boldsymbol{J}_j(p) = \left[\frac{\partial F(p)}{\partial z_j(p)}\right]. \tag{7}$$

Comparing (6) and (3), we can clearly see an equivalence between the quasi-Newton optimization process and the RLS algorithm employed by the NM-Delta

(under the foregoing assumptions). In consequence, the assumption $\partial^2 F/\partial z_i \partial z_j = \boldsymbol{I}$ greatly reduces the computational complexity and in fact is not that radical. Many algorithms exploit the assumption of the diagonal (not even block-diagonal) structure of the Hessian. The prefix 'quasi' in the name 'quasi-Newton' has been used not because of the simplifying assumptions, but because of the initialization technique used by the RLS process. The gradually-built estimation of the autocorrelation matrix inverse, $\boldsymbol{S}$, has to be initiated. The algorithm uses the initialization

$$\boldsymbol{S}(0) = \gamma^{-1}\boldsymbol{I}, \tag{8}$$

which is usually described as an equivalent to ensuring the well-conditioning of the estimated $\boldsymbol{A}$. It can be shown that such an initialization is equivalent to imposing the Tikhonov regularization on the LS problem solved (Hansen, 1993; Haykin, 1999). We will use this feature further on to adapt the γ values.

It is striking that the approach described above is very similar to the Levenberg-Marquardt algorithm. The L-M method has been derived for nonlinear models using the quadratic error function. For such models, the Hessian can be calculated according to the following formula:

$$\boldsymbol{H}(p) = \boldsymbol{J}(p)\boldsymbol{J}(p)^T + \boldsymbol{R}(p), \tag{9}$$

where $\boldsymbol{H}(p)$ is the Hessian, $\boldsymbol{J}(p)$ stands for an appropriately constructed matrix of gradients and $\boldsymbol{R}(p)$ denotes the term involving second-order derivatives, vanishing in the neighbourhood of the minimum. The Levenberg-Marquardt method calculates an estimate to the Hessian using the approximated formula

$$\hat{\boldsymbol{H}}(p) = \boldsymbol{J}(p)\boldsymbol{J}(p)^T + \alpha(p)\boldsymbol{I}, \tag{10}$$

where $\alpha(p)\boldsymbol{I}$ is the term representing an approximation of the second derivative, adapted in the course of learning. The L-M algorithm does not treat α as a regularization parameter and rather uses some heuristic of the 'search then converge' type to adapt its value. The L-M method starts with a value of α large enough so that the first term of (10) is dominated by the other. The optimization process in this phase is simply a gradient descent (cf. (3), taking into acount that the Hessian is 'constant'). In the course of the learning process, the algorithm decreases the α value, which 'activates' the first term of (10) and makes the other gradually vanish to zero, exactly the way in which $\boldsymbol{R}(p)$ in (9) behaves. Thus, in the course of learning, the value of the Hessian calculated with (10) converges to the true value of (9). The 'converge' phase involves calculating the product $\boldsymbol{J}(p)\boldsymbol{J}(p)^T$, which constitutes the main computational cost of the method.

4. (m, δ)-Adaptation Phase of the NM-Delta Algorithm

To relax the strong assumptions on the structure of $\partial^2 F/\partial z_j^2$, we can use an appropriate adaptation of the parameters m and δ. Intuitively, the idea is to 'stretch' the error surface so as to conform to the assumptions made on its curvature by the RLS

process. We can obtain the description of the error surface transformation taking place in the backpropagation process, as follows:

$$\frac{\partial^2 F}{\partial z_i \partial z_j} = \frac{\partial^2 F}{\partial u_i \partial u_j}\frac{\partial u_i}{\partial z_i}\frac{\partial u_j}{\partial z_j} + \frac{\partial^2 u_j}{\partial z_i \partial z_j}\frac{\partial F}{\partial u_j}. \tag{11}$$

In (11) there is a second-derivative factor $\partial^2 F/\partial u_i \partial u_j$. In order to calculate it, we can exploit the assumption on the diagonal structure of the matrix $[\partial F/\partial z^{l+1}]$ for the layer $l+1$ (in the earlier phase of the algorithm, care has already been taken to approximate the diagonal structure of that matrix). In such a case, we have

$$\frac{\partial^2 F}{\partial u_i^l \partial u_j^l} = \sum_m w_{mj}^{l+1} \sum_n w_{ni}^{l+1} \frac{\partial^2 F}{\partial z_n^{l+1} \partial z_m^{l+1}} = \sum_o w_{oj}^{l+1} w_{oi}^{l+1}, \tag{12}$$

with the last step of derivation using the assumed identity character of the $\partial^2 F/\partial z_n^{l+1} \partial z_m^{l+1}$ matrix. The other term of (11) is diagonal and non-zero for $i = j$. It is obvious that changing $\partial u_\alpha/\partial z_\alpha$ by means of the parameters m and δ we are not able to remove the off-diagonal terms of $\partial^2 F/\partial z_i \partial z_j$. We can only drive the (m, δ)-adaptation process in such a way that the diagonal terms be close to one. The necessary condition is

$$\frac{\partial^2 F}{\partial z_i^2} = \frac{\partial^2 F}{\partial u_i^2}\left(\frac{\partial u_i}{\partial z_i}\right)^2 + \frac{\partial^2 u_i}{\partial z_i^2}\frac{\partial F}{\partial u_i} = 1. \tag{13}$$

This expression is independent of the chosen error and activation functions.

In the case of the generalized logistic activation function used in the NM-Delta, we have

$$\begin{aligned} u_\alpha(z_\alpha) &= \frac{m_\alpha}{1 + e^{-\delta_\alpha z_\alpha}}, \\ \frac{\partial u_\alpha}{\partial z_\alpha} &= \frac{\delta_\alpha}{m_\alpha} u_\alpha (m_\alpha - u_\alpha), \\ \frac{\partial^2 u_\alpha}{\partial z_\alpha^2} &= \frac{\delta_\alpha}{m_\alpha} u_\alpha (m_\alpha - 2u_\alpha)\frac{\partial u_\alpha}{\partial z_\alpha}. \end{aligned} \tag{14}$$

An example of such a function with its first and second derivative is shown for reference in Fig. 1. Deriving from (13) the procedure of changing the values m and δ, we have made the following simplifying assumptions:

- It can be easily seen in Fig. 1 that there are conditions for which fulfilling (13) demands excessive values of δ_α/m_α (a very small value of $\partial F/\partial u_i$ in the saturation region of the activation function). We then impose some limits on the value of this ratio.
- We have decided to adapt only the m values, keeping δ constant.
- We have linearized the approximation function, i.e. we have neglected the other term of (13) involving the second derivative of the activation function. We can justify such an approach by choosing relatively small values of the parameters δ, which determine maximal curvatures of the activation functions.

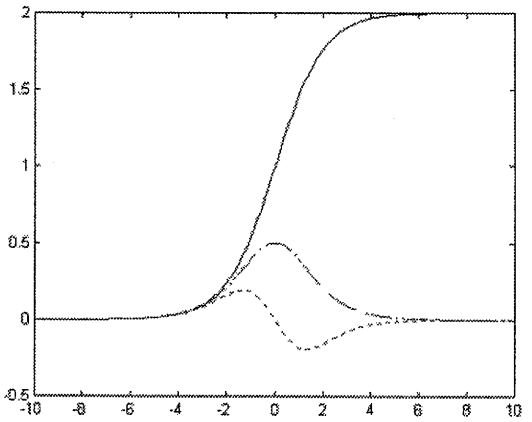


Fig. 1. Logistic activation function ($m = 2$, solid line) along with its first (dashed line) and second (dotted line) derivatives.

On the above assumptions, (13) reduces to

$$\frac{\partial^2 F}{\partial u_i^2}\left(\frac{\partial u_i}{\partial z_i}\right)^2 = 1, \tag{15}$$

which we use to calculate the m values in the subsequent epochs of the learning process.

5. Adaptation of the Regularization Parameter γ

As was pointed out, the initialization of the RLS algorithm is equivalent to the Tikhonov regularization (Tikhonov and Arsenin, 1977) of the least-squares problem. The LS problem with the Tikhonov regularization consists in finding

$$\min\left(\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 + \lambda^2 \|\boldsymbol{L}\,(\boldsymbol{x} - \boldsymbol{x}_0)\|_2\right). \tag{16}$$

The regularization parameter λ determines an equilibrium between the residual error $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$ measuring how well the model fits to the training data and the solution norm $\|\boldsymbol{L}(\boldsymbol{x} - \boldsymbol{x}_0)\|_2$ constraining large 'magnitudes' of $\boldsymbol{x}$ leading to non-smooth solutions. We often use $\boldsymbol{x}_0 = \boldsymbol{0}$ and $\boldsymbol{L} = \boldsymbol{I}$. It can be shown that

$$\gamma = \frac{1}{\lambda^2}. \tag{17}$$

The described regularization approach was identified to be equivalent with the weight decay technique (Bishop, 1995a), the learning with noise contamination (Bishop, 1995b), and it is also known as the ridge regression technique (Goutte, 1997). Selection of the regularization term was shown to be equivalent to the choice of priors in Bayesian inference (Girosi, 1995).

There are various methods of the optimal choice of the regularization parameter λ. The best known approach is to choose the value that minimizes the generalization error. This error, in turn, is estimated by means of the crossvalidation or by using the so-called algebraic estimator (Goutte, 1997). Though regularization should be considered as a global, data-driven approach which aims at improving generalization, we will apply it in a local perspective, on the basis of partial RLS problems solved, to each layer of the network separately. Such an approach is supported by the tradition of the least-squares regularization (see e.g. Hansen, 1994). The method of choice for our problem is the technique proposed by Hansen *et al.* (1994), see also (Larsen *et al.*, 1997; Moody, 1991). The method uses optimization of the generalization error estimate E_{tst} w.r.t. the regularization parameters. The method uses the so-called FPER (Final Prediction Error for models with Regularization) algebraic estimator proposed in (Larsen and Hansen, 1994):

$$\text{FPER} = \frac{N + \hat{m}_2}{N - 2\hat{m}_1 + \hat{m}_2} C_N(\hat{\boldsymbol{w}}), \tag{18}$$

where N is the number of training samples, $\hat{\boldsymbol{w}}$ is the estimated weight vector minimizing the regularized cost function value after the presentation of N samples, C_N is the mean-square cost and m_1, m_2 are two different effective numbers of weights, respectively defined as

$$\begin{aligned} \hat{m}_1 &= \operatorname{tr}\left[\boldsymbol{H}_N\left(\hat{\boldsymbol{w}}\boldsymbol{K}_N^{-1}(\hat{\boldsymbol{w}})\right)\right], \\ \hat{m}_2 &= \operatorname{tr}\left[\boldsymbol{H}_N\left(\hat{\boldsymbol{w}}\boldsymbol{K}_N^{-1}(\hat{\boldsymbol{w}})\right)\boldsymbol{H}_N\left(\hat{\boldsymbol{w}}\boldsymbol{K}_N^{-1}(\hat{\boldsymbol{w}})\right)\right], \end{aligned} \tag{19}$$

where $\boldsymbol{H}_N$ is the Hessian matrix of the mean-square cost function w.r.t. the weights and $\boldsymbol{K}_N$ is the Hessian of the error function augmented by the regularization term. The values of m_1 and m_2 depend on the regularization parameters. A matrix estimate of $\boldsymbol{K}_N^{-1}$ is readily available in each epoch as the matrix $\boldsymbol{S}$ (as has been shown in Section 3). As for $\boldsymbol{H}_N$, we use the diagonal approximation applied in the method of (Hansen *et al.*, 1994), known from the OBD algorithm:

$$\frac{\partial^2 F}{\partial u_j^2} \approx \left(\frac{\partial F}{\partial u_j}\right)^2. \tag{20}$$

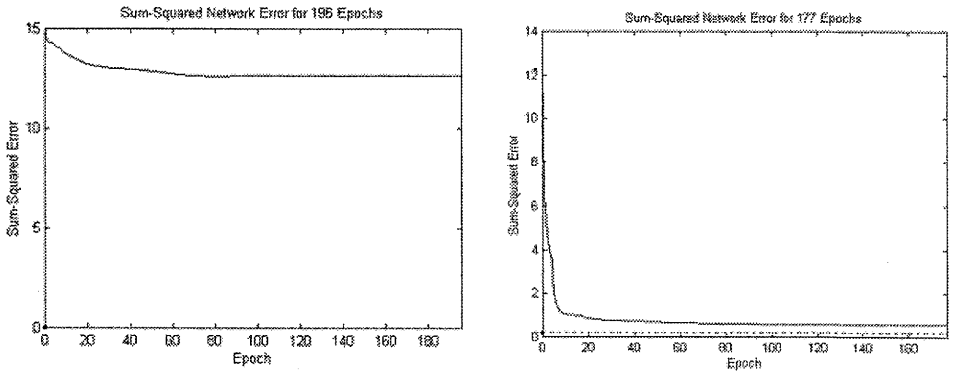
In order to optimize the generalization error estimate, we use a simple evolutionary strategy of reversing the regularization parameter changes when the generalization error estimate increases.

6. Simulations Results

The learning problem for simulations has been drawn from the DELVE (Data for Evaluating Learning in Valid Experiments) repository, held at the University of Toronto, `www.cs.toronto.edu/~delve/`. The motivation for the project is to provide uniform framework to report a performance of the learning algorithm. Following the authors' intentions, we used one of the so-called development datasets, namely the kin8nm

dataset. The set provides synthetically generated data from a realistic simulation of the forward kinematics of an eight-link all-revolute robot arm. The data has eight input channels, corresponding to the angular positions of the eight joints of the robot arm and one output referring the Cartesian distance of the end-point of the arm from the position (0.1, 0.1, 0.1). The transfer function is highly nonlinear. The data are contaminated with the noise of a medium level. For the training we used 1000 examples from the dataset. All simulations were conducted using Matlab software. The Levenberg-Marquardt implementation comes from the Neural Network Toolbox, and the NM-Delta implementation is of our own.

Figure 2 draws a comparison of the learning of the kin8nm problem by the original NM-Delta and the Levenberg-Marquardt algorithm. The regularization parameter γ was chosen arbitrarily and probably not very well, because the performance of the network is poor. The results are the best from among a few guesses for the values of the regularization parameter and the learning speed for the GDM phase. They depict troubles we have while tuning the learning parameters.



(a) (b)

Fig. 2. The training error evolution in the NM-Deltanetwork (a) and in the network using the Levenberg-Marquardt algorithm (b).

Figure 3 shows results of the learning process for the NM-Delta network with an automatic choice of m parameters, as described in Section 4. There is no significant difference between the processes. This suggests that the original GDM algorithm conforms in some way to the assumptions on the Hessian structure stated in Section 3. Certainly, some further investigation is needed here. The results of the simulations suggest that apparently the main obstacle to achieve a better fit are the stiff values of the regularization parameters bearing false assumptions on the smoothness of the approximated function.

Figure 4 shows the results of the learning process for the NM-Delta network using the adaptation of the regularization parameters and for the network using the Levenberg-Marquardt method. As can be seen, the barrier of improper regularization assumptions has been broken. The Levenberg-Marquardt still shows better perfor-

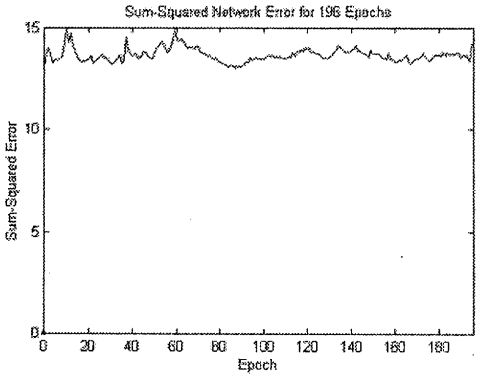
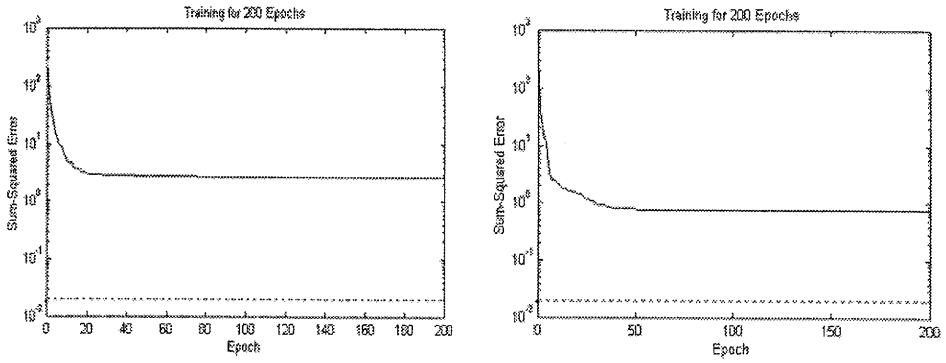


Fig. 3. The training error evolution in the NM-Delta network with the adaptation of m parameters described in Section 4.



(a) (b)

Fig. 4. The training error evolution in the NM-Delta network with adapted regularization parameters (a) and in the network trained with the Levenberg-Marquardt algorithm (b).

mance. It reaches a target mean square error of ca 0.74 after 200 learning epochs as opposed to ca 3.40 achieved by the NM-Delta The learning is also somewhat slower in the initial phase for the NM-Delta than for the L-M. The difference comes undoubtedly from the approximation to the Hessian structure made by the NM-Delta algorithm, more coarse than that for the Levenberg-Marquardt. Besides, the choice of the learning speed of the regularization parameter adaptation exerts influence on the error level achieved. However, the computational cost comparison reveals a clear advantage of the NM-Delta network. Matlab simulation for the learning process under consideration takes 2.7890e+9 Flops, while for the NM-Delta it is 0.4417e+9. And of course, the Levenberg-Marquardt algorithm scales in a much worse way. The Hessian evaluation itself needs $\mathcal{O}(N^2)$ operations, where N is the number of weights. For the NM-Delta, N is the sum of the number of inputs to the network and the number of hidden neurons.

7. Conclusions

The paper presents the approach that is rather far from the intentions of regularization theory, especially in its Bayesian context. We have tried to use regularization techniques for tuning parameters of the optimization process. The dependence of the optimization process on the curvature of the error surface is an unquestionable postulate exploited more or less extensively in many algorithms. On the other hand, the act of the regularization is motivated by some assumptions on the features of the approximated function, and thus—on the error surface. We have tried to get something from both the worlds. We think that such an approach can be fruitful in the analysis of the learning algorithms.

Acknowledgments

This work was supported by the State Committee for Scientific Research under grant No. 8T11C 022 18.

References

Bishop C.M. (1995a): *Neural Networks for Pattern Recognition.* — Oxford: Clarendon Press.

Bishop C.M. (1995b): *Training with noise is equivalent to Tikhonov regularization.* — Neural Comput., Vol.7, No.1, pp.110–116.

Girosi F., Jones M. and Poggio T. (1995): *Regularization Theory and Neural Networks Architectures.* — Neural Comput., Vol.7, No.2, pp.219–269.

Gołąbek P., Kosiński W. and Weigl M. (1999): *Adaptation of learning rate via adaptation of weight vector in modified M-Delta networks*, In: Computational Intelligence and Applications (P.S. Szczepaniak, Ed.). — Heidelberg, New York: Physica-Verlag pp.156–163,

Goutte C. (1997): *Statistical learning and regularisation for regression.* — Ph.D. Dissertation, L'Université Paris 6.

Hansen P.C. (1994): *Regularization tools: A Matlab package for analysis and solution of discrete ill-posed problems.* — Numer. Algorithms, No.6, pp.44–55.

Hansen P.C. and O'Leary D.P. (1993): *The use of the L-curve in the regularization of discrete ill-posed problems.* — SIAM J. Sci. Comput., No.14, pp.1487–1503.

Hansen L.K., Rasmussen C.E., Svarer C. and Larsen J. (1994): *Adaptive regularization.* — Proc. IEEE Workshop *Neural Networks for Signal Processing IV*, Piscataway, New Jersey, pp.78–87.

Haykin S. (1996): *Adaptive Filter Theory.* — New Jersey: Prentice Hall Int.

Haykin S. (1999): *Neural Networks—A Comprehensive Foundation.* — New Jersey: Prentice Hall.

Kosiński W. and Weigl M. (1996): *Approximation by a neural network with modified activation functions.* — Proc. 3rd Biennial Europeean Engineering System Design & Analysis Conference (ESDA), Vol.7, Montpellier, France.

Kosiński W. and Weigl M. (1998): *General mapping approximation problems solving by neural and fuzzy inference networks.* — SAMS, Vol.30, No.1, pp.11–28.

Krushke J.K. and Movellan J.R. (1991): *Benefits of Gain: Speeded Learning and Minimal Hidden Layers in Backpropagation Networks.* — IEEE Trans. Syst. Man Cybern., Vol.21, No.1, pp.273–280.

Larsen J. and Hansen L.K. (1994): *Generalization performance of regularized neural network models.* — Proc. IEEE Workshop *Neural Networks for Signal Processing IV*, Piscataway, New Jersey, pp.42–51.

Larsen J., Svarer C., Nonboe Andersen L. and Hansen L.K. (1997): *Adaptive regularization in neural network modelling (preprint).* — Available at ftp://eivind.imm.dtu.dk/dist/1997/larsen.bot.ps.Z

Moody J.E. (1991): *Note on generalization, regularization and architecture selection in nonlinear learning systems.* — Proc. 1991 IEEE-SP Workshop, Piscataway, New Jersey, pp.1–10.

Rutkowski L. (1994): *Adaptive Filters and Adaptive Signal Processing.* — Warsaw: WNT (in Polish).

Thimm G., Moerland P. and Fiesler E. (1996): *The interchangeability of learning rate and gain in backpropagation neural networks.* — Neural Comput., Vol.8, No.2, pp.451–460.

Tikhonov A.N. and Arsenin V.Y. (1977): *Solutions of Ill-Posed Problems* — New York: Wiley.

Weigl M. (1995): *Neural networks and fuzzy inference systems for approximation problems.* — Ph.D. Thesis, Inst. Fundamental Technological Research, Warsaw, Poland (in Polish).