Trends and Applications in Constructive Approximation
(Eds.) M.G. de **Bruin**, D.H. **Mache** & J. **Szabados**
International Series of Numerical Mathematics Vol. 1??

Rprop Using the Natural Gradient

Christian Igel Marc Toussaint Wan Weishui

Abstract

Gradient-based optimization algorithms are the standard methods for adapting the weights of neural networks. The natural gradient gives the steepest descent direction based on a non-Euclidean, from a theoretical point of view more appropriate metric in the weight space. While the natural gradient has already proven to be advantageous for online learning, we explore its benefits for batch learning: We empirically compare Rprop (resilient backpropagation), one of the best performing first-order learning algorithms, using the Euclidean and the non-Euclidean metric, respectively. As batch steepest descent on the natural gradient is closely related to Levenberg-Marquardt optimization, we add this method to our comparison.

It turns out that the Rprop algorithm can indeed profit from the natural gradient: the optimization speed measured in terms of weight updates can increase significantly compared to the original version. Rprop based on the non-Euclidean metric shows at least similar performance as Levenberg-Marquardt optimization on the two benchmark problems considered and appears to be slightly more robust. However, in Levenberg-Marquardt optimization and Rprop using the natural gradient computing a weight update requires cubic time and quadratic space. Further, both methods have additional hyperparameters that are difficult to adjust. In contrast, conventional Rprop has linear space and time complexity, and its hyperparameters need no difficult tuning.

1 Introduction

Artificial neural networks such as Multi-Layer Perceptrons (MLPs) have become standard tools for regression. In essence, an MLP with fixed structure defines a differentiable mapping from a parameter space $\mathbb{R}^n$ to the space of functions. Although this mapping is typically not surjective, one can prove that in principle every continuous function can be well approximated (e.g., see [8], the upper bound

on the approximation error depends on the network structure). For an MLP with fixed structure, a regression problem reduces to the adaptation of the parameters, the weights, of the network given the sample data. This process is usually referred to as learning.

Since MLPs are differentiable, gradient-based adaptation techniques are typically applied to adjust the weights. The earliest and most straightforward adaptation rule, ordinary gradient descent, adapts weights proportional to the partial derivatives of the error functional [1, 17]. Several improvements of this basic adaptation rule have been proposed, some of them based on elaborated heuristics, others on theoretical reconsideration of gradient-based learning. Here we consider three of them, natural gradient descent, resilient backpropagation, and Levenberg-Marquardt optimization. We combine ideas of all three approaches to a new method we call *natural Rprop*.

Resilient backpropagation (Rprop, [10, 15, 16]) is a well-established modification of the ordinary gradient descent. The basic idea is to adjust an individual step size for each parameter to be optimized. These step sizes are not proportional to the partial derivatives but are themselves adapted based on some heuristics. Ordinary gradient descent computes the direction of steepest descent by implicitly assuming a Euclidean metric on the weight space. However, as we are interested in the function corresponding to a weight configuration, a more appropriate metric would take distances in the function space into account. Amari [2] proposed to make use of methods from differential geometry to determine the steepest descent direction, the negative *natural gradient*, based on such a non-Euclidean metric.

Our investigation is based on the following idea: If the natural gradient points in a better descent direction and Rprop improves the ordinary gradient descent, can a combination of both methods profit from the advantages of the two approaches? Can we get best of both worlds, the increased robustness of Rprop and the improved convergence speed due to the decoupling of weight interdependencies when using the right metric?

In order to asses the performance of the new learning approach, we compare it to a standard Rprop algorithm and to Levenberg-Marquardt optimization [12, 13]. We choose Levenberg-Marquardt optimization, because in the function regression scenario that we consider it turns out that natural gradient descent is closely related to Levenberg-Marquardt optimization. In addition, we borough ideas from this classical technique to increase the robustness of the calculation of the steepest descent direction within natural Rprop.

In the next section we give brief but comprehensive descriptions of Rprop, Levenberg-Marquardt optimization, and the natural gradient. Then we introduce their combination, the natural Rprop. Section 3 presents experimental results on two benchmark problems. Since the Levenberg-Marquardt algorithm as well as the natural Rprop introduce new hyperparameters, we particularly look at the robustness of the algorithms in terms of the choices of these parameters. We conclude by discussing the results in section 4.

2 Background

We understand a feed-forward MLP with n_{in} inputs and n_{out} outputs as a function $f(\,\cdot\,; w) : \mathbb{R}^{n_{\text{in}}} \to \mathbb{R}^{n_{\text{out}}}$ describing some input-output behavior. The function is parameterized by a weight vector $w \in W = \mathbb{R}^n$ as indicated by the notation. For instance, $y = f(x; w)$ is the output of the network for a given input x and weights w. To address the components of the weight vector w we use upper indices $(w^1, .., w^n)$ to clarify that they are contra-variant and not co-variant (this will become more apparent and relevant when describing the natural gradient). As an error measure $E(w)$ to be minimized we assume, in the following, the mean squared error (MSE) on a batch $\{(x_1, y_1), .., (x_P, y_P)\} \in (\mathbb{R}^{n_{\text{in}}} \times \mathbb{R}^{n_{\text{out}}})^P$ of sample data points. For simplicity, we restrict ourselves to $n_{\text{out}} = 1$ and define

$$E(w) = \frac{1}{P} \sum_{p=1}^{P} ||f(x_p; w) - y_p||^2 \ .$$

In the remainder of this section, we describe four methods for gradient-based minimization of $E(w)$: we review Rprop, natural gradient descent, and Levenberg-Marquardt optimization, and introduce a new approach, natural Rprop.

2.1 Resilient Backpropagation

The Rprop algorithms are among the best performing first-order batch learning methods for neural networks with arbitrary topology [9, 10, 15, 16]. They are

- very fast and accurate (e.g., compared to conjugate gradient methods, Quickprop etc.),
- very robust in terms of the choices of their hyperparameters,
- first-order methods, therefore time and space complexity scales linearly with the number of parameters to be optimized,
- only dependent on the sign of the partial derivatives of the objective function and not on their amount, therefore they are suitable for applications where the gradient is numerically estimated or the objective function is noisy, and
- easy to implement and not very sensitive to numerical problems.

In the following, we describe the Rprop variant with improved backtracking introduced in [9]. The Rprop algorithms are iterative optimization methods. Let t denote the current iteration (epoch). In epoch t, each weight is changed according to

$$w^i(t+1) = w^i(t) - \text{sign}\left(\frac{\partial E(t)}{\partial w^i}\right) \cdot \Delta^i(t) \ .$$

The direction of the change depends on the sign of the partial derivative, but is independent of its amount. The individual step sizes $\Delta^i(t)$ are adapted based on changes of sign of the partial derivatives of $E(w)$ w.r.t. the corresponding weight:

If $\frac{\partial E(t-1)}{\partial w^i} \cdot \frac{\partial E(t)}{\partial w^i} > 0$ then $\Delta^i(t)$ is increased by a factor $\eta^+ > 1$, otherwise $\Delta^i(t)$ is decreased by multiplication with $\eta^- \in]0,1[$. Additionally, some Rprop methods implement weight-backtracking. That is, they partially retract "unfavorable" previous steps. Whether a weight change was "unfavorable" is decided by a heuristic. We use an improved version of the original algorithm called iRprop$^+$, which is described in pseudo-code in Table 1. The difference compared to the original Rprop proposed in [16] is that the weight-backtracking heuristic considers both the evolution of the partial derivatives and the overall error. For a comparison of iRprop$^+$ with other Rprop variants and a detailed description of the algorithms the reader is referred to [10].

for each w^i **do**
  if $\frac{\partial E(t-1)}{\partial w^i} \cdot \frac{\partial E(t)}{\partial w^i} > 0$ **then**
    $\Delta^i(t) = \min\left(\Delta^i(t-1) \cdot \eta^+, \Delta_{\max}\right)$
    $w^i(t+1) = w^i(t) - \operatorname{sign}\left(\frac{\partial E(t)}{\partial w^i}\right) \cdot \Delta^i(t)$
  elseif $\frac{\partial E(t-1)}{\partial w^i} \cdot \frac{\partial E(t)}{\partial w^i} < 0$ **then**
    $\Delta^i(t) = \max\left(\Delta^i(t-1) \cdot \eta^-, \Delta_{\min}\right)$
    if $E(t) > E(t-1)$ **then** $w^i(t+1) = w^i(t-1)$
    $\frac{\partial E(t)}{\partial w^i} := 0$
  elseif $\frac{\partial E(t-1)}{\partial w^i} \cdot \frac{\partial E(t)}{\partial w^i} = 0$ **then**
    $w^i(t+1) = w^i(t) - \operatorname{sign}\left(\frac{\partial E(t)}{\partial w^i}\right) \cdot \Delta^i(t)$
  fi
od

Table 1: The iRprop$^+$ algorithm with improved weight-backtracking scheme as proposed in [10].

2.2 Levenberg-Marquardt Optimization

Levenberg-Marquardt optimization [6, 12, 13] is based on the idea that, to minimize the error functional $E(w)$, one should find weights such that the derivatives $\frac{\partial E(w)}{\partial w^i}$ vanish. This search can be realized with a Newton step on an approximation of the error functional as follows. Consider the linear approximation of $f(x;w)$

around $w(t)$,

$$\hat{f}(x;w) = f(x;w(t)) + \sum_{j=1}^{n}[w^j - w^j(t)]\,\frac{\partial f(x;w(t))}{\partial w^j}\ .$$

Substituting $\hat{f}$ for f in the MSE gives a new error function $\hat{E}(w)$ with gradient

$$\begin{aligned}\frac{\partial \hat{E}(w)}{\partial w^i} &= \frac{2}{P}\sum_{p=1}^{P}[\hat{f}(x_p;w) - y_p]\,\frac{\partial \hat{f}(x_p;w)}{\partial w^i} \\ &= \frac{\partial E(w(t))}{\partial w^i} + \sum_j A_{ij}(w(t))\ [w^j - w^j(t)]\ . \end{aligned} \tag{1}$$

Here the $n \times n$ matrix $A_{ij}(w)$ has the entries

$$A_{ij}(w) = \frac{2}{P}\sum_{p=1}^{P}\frac{\partial f(x_p;w)}{\partial w^i}\frac{\partial f(x_p;w)}{\partial w^j}\ .$$

Setting (1) to zero (i.e., doing a Newton step on $\hat{E}$) leads to the weight update

$$w^i(t+1) = w^i(t) - \sum_{j=1}^{n} A^{ij}(w(t))\ \frac{\partial E(w(t))}{\partial w^j}\ . \tag{2}$$

Here A^{ij} is the inverse matrix of A_{ij}. This weight update would lead to an optimum if $\hat{E}(w) = E(w)$. This is in general not the case and the weight update rule (2) is only reasonable close to a minimum. Therefore, the idea is to automatically blend between (2) and standard steepest descent:

$$w^i(t+1) = w^i(t) - \sum_{j=1}^{n}[A_{ij} + \lambda \mathbf{I}_{ij}]^{-1}\,\frac{\partial E(w(t))}{\partial w^j}\ ,$$

where the parameter $\lambda > 0$ allows soft switching between the two strategies. A large λ corresponds to simple gradient descent. There are several heuristics to adapt λ. We use the most common one to decrease λ by multiplication with $\lambda_- \in]0,1[$ if the error decreased, and to increase it by multiplication with $\lambda_+ > 1$ (usually $\lambda_- = \lambda_+^{-1}$), otherwise. A drawback of Levenberg-Marquardt optimization is that the choice of λ_0 (the initial value for λ), λ_-, and λ_+ is crucial for the performance of the algorithm.

2.3 Natural Gradient Descent

Basically, natural gradient descent is steepest descent with a non-Euclidean metric on the parameter space. Two simple facts motivate the natural gradient: First, the

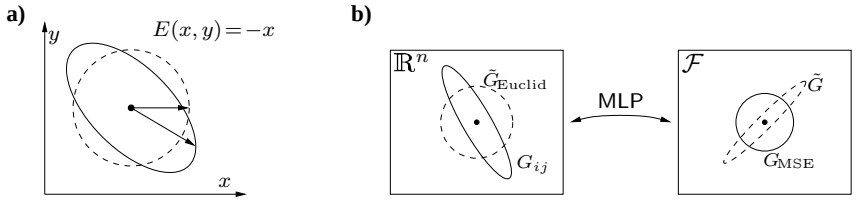

Figure 1: **a)** The steepest descent direction of a functional $E(x, y)$ depends on the metric: The ellipses mark the set of vectors of unit length for the Euclidean metric (dashed circle) and the metric $G = \begin{pmatrix} 3/4 & 1/4 \\ 1/4 & 3/4 \end{pmatrix}$ (ellipse). For the ellipse, the unit length vector that decreases $E(x, y) = -x$ the most is not pointing directly to the right, it is generally given by equation (3). **b)** An MLP establishes a differentiable relation between the weight space $W = \mathbb{R}^n$ and the manifold of functions $\mathcal{F}$. A canonical distance measure G_{MSE} on the function space induces a non-Euclidean metric G_{ij} on the weight space w.r.t. which steepest descent should be performed. In contrast, if a Euclidean metric $\tilde{G}_{\mathrm{Euclid}}$ is presumed on the weight space, this generally leads to a "non-diagonal" metric $\tilde{G}$ on the function space. Typically, a non-diagonal metric on the function space is undesirable because this leads to negative inference and cross-talk; e.g., during online learning, if one functional components is trained, others are untrained according to the off-diagonal entries of the metric [18]. Using the natural gradient avoids this effect of catastrophic forgetting during online learning.

steepest descent direction generally depends on the choice of metric on the parameter space—this is very often neglected in standard textbooks describing gradient descent. See figure 1a) for an illustration. Second, there are good arguments to assume a non-Euclidean metric on the parameter space: Generally, there exists no a priori reason why the Euclidean metric on the parameter space should be a preferential distance measure between solutions. In fact, in the case of function regression, one typically assumes a canonical distance measures *on the function space*, like the mean squared error, or a likelihood measure on the space of distributions, which translate to non-trivial metrics on the parameter space, see figure 1b). Amari [2, 5, 4] was the first to realize the implications of these facts in the case of gradient-based adaptation of MLPs. In the following, we give a simple derivation of the natural gradient.

An MLP represents a differentiable mapping f from the parameter space $W = \mathbb{R}^n$ to the manifold $\mathcal{F}$ of functions. We write $f : \ w \in W \mapsto f(\,\cdot\,; w) \in \mathcal{F}$. Let d be a distance measure on $\mathcal{F}$. Here we assume that $d(f, h)$ is the mean squared

distance on a batch $\{x_1, .., x_P\}$ of data points between two functions f and h,

$$d(f,h) = \frac{1}{P}\sum_{p=1}^{P}[f(x_p) - h(x_p)]^2 \ .$$

The pull-back of this metric onto the parameter space is, by definition,

$$G_{ij}(w) = \frac{1}{P}\sum_{p=1}^{P}\frac{\partial f(x_p;w)}{\partial w^i}\frac{\partial f(x_p;w)}{\partial w^j} \ .$$

The meaning of this metric G_{ij} on W is: if we measure distances in W using G_{ij}, then these distances are guaranteed to be equal to the mean squared distance when measured on the function space $\mathcal{F}$.[1] Further, if we determine the steepest descent direction in W using the metric G_{ij} we can be sure to find the direction in which the mean squared error on $\mathcal{F}$ decreases fastest—which is generally not true when using the Euclidean metric!

The steepest descent direction of a functional $E(w)$ over W is given as the vector δ with components

$$\delta^j = \sum_{i=1}^{n} G^{ij}(w)\,\frac{\partial E(w)}{\partial w^i} \ . \tag{3}$$

Here G^{ij} is the inverse matrix of G_{ij}. (Upper indices denote so-called contravariant components.) Thus, in summary, natural gradient descent with learning rate $\eta > 0$ reads

$$w^i(t+1) = w^i(t) - \eta\sum_{j=1}^{n} G^{ij}(w(t))\,\frac{\partial E(w(t))}{\partial w^j} \ . \tag{4}$$

There exists an online version of the natural gradient [5] that approximates the inverse natural metric G^{ij} on the fly and reduces the negative effects of co-inference during online learning (cf. figure 1b).

Comparing Levenberg-Marquardt adaptation (2) with batch natural gradient (4) descent we find that for $\lambda = 0$ they are equivalent since $A_{ij} = G_{ij}$ (in the case of the mean squared distance $d(f,h)$ on $\mathcal{F}$). This fact has previously been observed by [3, 7, 11]. A small difference is the robustness term for $\lambda \neq 0$. Note that for different distance measures on $\mathcal{F}$, generally $A_{ij} = G_{ij}$ does not hold.

2.4 Natural Rprop

Rprop is a batch gradient-based learning algorithm that overcomes the problems of standard gradient descent by automatically adjusting individual step sizes. The

[1] More precisely, if we measure the distance between w_1 and w_2 in W by the length of the geodesic w.r.t. $G_{ij}(w)$, then this distance is guaranteed to be equal to the mean squared distance $d(f_1, f_2)$ between the two corresponding functions $f_1 = f(\,\cdot\,; w_1)$ and $f_2 = f(\,\cdot\,; w_2)$ in $\mathcal{F}$.

	space	time
iRprop$^+$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Levenberg-Marquardt optimization	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$
natural gradient descent / natural iRprop$^+$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$

Table 2: The complexity of the three algorithms w.r.t. space and time. The number of weights in the MLP is denoted by n.

natural gradient points in a direction that is more appropriate for steepest descent optimization. Now, the question arises whether it can be beneficial to combine natural gradient descent with the heuristics of Rprop. Recall that one of the main features of Rprop is that the update step sizes depend only on the signs of the gradient. Since the metric G and also its inverse G^{-1} are always positive definite, a vector transformed by G changes its direction by up to 90°. The angle between ordinary and natural gradient descent directions can also be up to 90° (which also becomes apparent from figure 1a). Thus, the signs can generally change when replacing the ordinary gradient by the natural gradient in the Rprop algorithm and, therefore, adaptation behavior changes.

We hence propose to combine iRprop$^+$, the natural gradient, and the robustness term $\lambda \mathbf{I}_{ij}$ of Levenberg-Marquardt simply by replacing the ordinary gradient $\frac{\partial E(w)}{\partial w^i}$ by the robust natural gradient

$$\left[G_{ij} + \lambda \operatorname{trace}(G_{ij})\, \mathbf{I}_{ij}\right]^{-1} \frac{\partial E(w)}{\partial w^i}$$

within the iRprop$^+$algorithm. As in Levenberg-Marquardt optimization, the parameter $\lambda \in \mathbb{R}^+$ blends between the natural gradient and the ordinary gradient by adding a weighted unity matrix $\mathbf{I}_{ij}$. Additionally, we weight the term proportional to the trace of G such that the blending becomes relative w.r.t. the orders of the Eigenvalues of G. We use the same update rule as before, λ is reduced by multiplication with $\lambda_- \in]0,1[$ if the error decreased and is set to $\lambda \leftarrow \lambda \cdot \lambda_+$, otherwise (usually $\lambda_- = \lambda_+^{-1}$). We call this new algorithm *natural Rprop*.

Table 2 displays the complexity of iRprop$^+$, the Levenberg-Marquardt algorithm, and natural gradient descent / natural Rprop w.r.t. space and time. Both, Levenberg-Marquardt and the natural gradient require the storage and inversion of a $n \times n$-matrix, with n being the number of weights, and this dominates the cost of these algorithms leading to cubic time complexity and quadratic space complexity. In contrast, iRprop$^+$needs only linear time and space to update weight by weight separately and to store the weights and the step size for each weight.

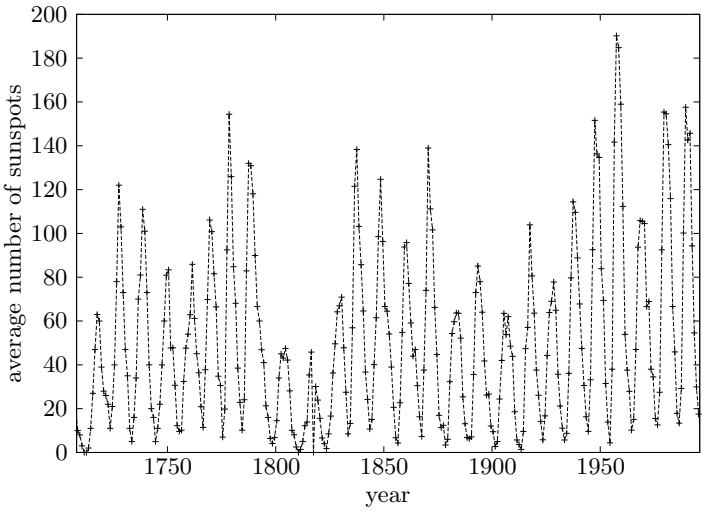

Figure 2: Time series of average number of sunspots observed per year.

3 Experiments

First, the two benchmark problems, sunspot prediction and extended XOR, are introduced. Then we describe the experimental setup. Finally, the empirical results are presented.

3.1 Benchmark Problems

The goal of the sunspot prediction task is to reproduce the time series of the average number of sunspots observed per year, see figure 2. The data are available from `http://sidc.oma.be`. The average number of spots from the years $t-1$, $t-2$, $t-4$, and $t-8$ are given to predict the value for the year t. The training set contains 289 patterns. The first year to predict is 1708. The input values are normalized between 0.2 and 0.8.

The extended XOR task, see figure 3, is an artificial classification benchmark [14]. The 1800 training patterns $(x, c) \in \mathbb{R}^2 \times \{0, 1\}$ are sampled from

$$p(x,c) = \frac{1}{|C_0| + |C_1|} \sum_{\mu \in C_c} \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/(2\sigma^2)} \ ,$$

with $C_0 = \{(1,5),(5,5),(3,3),(1,1),(5,1)\}$, $C_1 = \{(3,5),(1,3),(5,3),(3,1)\}$, and variance $\sigma^2 = 0.2$.

We want to evaluate the optimization speed of different learning algorithms. Thus, in both benchmark problems we just consider the task of learning the sample data and do not consider the import issue of generalization.

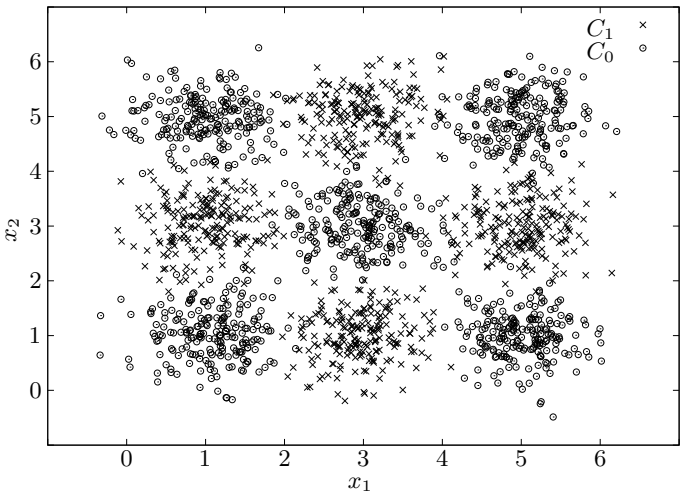

Figure 3: The extended XOR problem.

3.2 Experiments

We compare the standard iRprop$^+$, natural iRprop$^+$ (i.e., iRprop$^+$ using the natural gradient), and LevenbergMarquardt optimization on the two benchmark problems. The Rprop parameters are set to default values $\eta^+ = 1.2$, $\eta^- = 0.5$, $\Delta_{\min} = 0$, $\Delta_{\max} = 50$, and $\Delta_0 = 0.01$. For natural iRprop$^+$ and Levenberg-Marquardt optimization, we test all combinations of $\lambda_0 \in \{0.01, 0.1, 1, 10\}$ and $\lambda_+^{-1} = \lambda_- \in \{0.5, 0.6, \ldots, 0.9\}$. For every optimization method and parameter setting 20 trials starting from random initializations of the weights are performed; the 20 different initializations are the same for every algorithm and parameter combination. That is, a total of 840 NNs are trained. The number of iterations (learning epochs) is set to 5000.

For the sunspot prediction problem, a 4-10-1 NN architecture without shortcut connections is chosen. The 10 hidden neurons have sigmoidal transfer functions, the logistic / Fermi function $f(x) = 1/(1 + e^{-x})$, and the output neuron is linear. For the extended XOR, we use a 2-12-1 architecture without shortcut connections and only sigmoidal transfer functions. These architectures have not been tailored to the problems (and hence the absolute results are far from being optimal).

3.3 Results

The results are summarized in figures 4 and 5: Shown are the error trajectories for the 20 parameter combinations averaged over the 20 trials for Leven-

berg-Marquardt optimization and natural iRprop$^+$, respectively. The parameter settings yielding the lowest final error on average on the extended XOR problem were $\lambda_0 = 0.01$ and $\lambda_- = 0.5$ for Levenberg-Marquardt optimization and $\lambda_0 = 0.1$ and $\lambda_- = 0.8$ for natural iRprop$^+$. On the the sunspot prediction task, the best results were obtained for $\lambda_0 = 10, \lambda_- = 0.9$ and $\lambda_0 = 0.01, \lambda_- = 0.8$, respectively. The results corresponding to these parameter settings are compared to the standard Rprop in the lowest plots in figures 4 and 5. The differences between the error values of those three curves in the final iteration are pairwise statistically significant (Wilcoxon rank sum test, $p < .001$). The results show:

- The performance of natural iRprop$^+$ and LevenbergMarquardt optimization strongly depends on the choice of λ_0 and λ_-. The "best" values for λ_0 and λ_- are task-dependent. However, natural iRprop$^+$ appears to be more robust.

- For an appropriate parameter setting, Levenberg-Marquardt optimization and natural iRprop$^+$ clearly outperform the standard iRprop$^+$. However, the latter has lower computational complexity and does not depend on critical parameters such as λ_0 and λ_-.

- For one task, the Levenberg-Marquardt method yielded the best final solutions (averaged over 20 trials), for the other iRprop$^+$ using the natural gradient. In both problems, the iRprop$^+$ combined with the natural gradient seems to be slightly faster in the early iterations.

4 Conclusions

In this study, we compared Levenberg-Marquardt optimization (which can be regarded as some kind of batch natural gradient learning in our scenario), iRprop$^+$, and iRprop$^+$ using the natural gradient (natural iRprop$^+$) for optimizing the weights of feed-forward neural networks. It turned out that the Rprop algorithm can indeed profit from using the natural gradient, although the updates done by Rprop are not collinear with the (natural) gradient direction. Natural iRprop$^+$ shows similar performance as Levenberg-Marquardt optimization on two test problems. The results indicate that natural iRprop$^+$ is a little bit faster in the early stages of optimization and more robust in terms of the choices of the parameters λ_0, λ_-, and λ_+. However, a more extensive empirical investigation is needed to substantiate these findings. The standard iRprop$^+$ algorithm is slower than the other methods with appropriate parameters for λ_0, λ_-. Still, these parameters are problem dependent, they considerably influence the performance of the methods, and they are not easy to adjust. Further, the computational costs of each optimization step grow from linear to cubic when replacing Rprop with one of the other two methods. Hence, we conclude that some Rprop algorithm is still the batch-learning method of choice.

Acknowledgments

This work was supported by the DFG, grant Solesys-II SCHO 336/5-2.

References

[1] S. Amari. A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, 16(3):299–307, 1967.

[2] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276, 1998.

[3] S. Amari and S. C. Douglas. Why natural gradient? In *Proceedings of the 1998 IEEE International Conference Acoustics, Speech, Signal Processing (ICASSP 1998)*, volume II, pages 1213–1216, 1998.

[4] S. Amari and H. Nagaoka. *Methods of Information Geometry*. Number 191 in Translations of Mathematical Monographs. American Mathematical Society and Oxford University Press, 2000.

[5] S. Amari, H. Park, and K. Fukumizu. Adaptive method of realizing natural gradient learning for multilayer perceptrons. *Neural Computation*, 12:1399–1409, 2000.

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

[7] T. Heskes. On "natural" learning and pruning in multi-layered perceptrons. *Neural Computation*, 12(4):881–901, 2000.

[8] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[9] C. Igel and M. Hüsken. Improving the Rprop learning algorithm. In H. Bothe and R. Rojas, editors, *Proceedings of the Second International ICSC Symposium on Neural Computation (NC 2000)*, pages 115–121. ICSC Academic Press, 2000.

[10] C. Igel and M. Hüsken. Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing*, 50(C):105–123, 2003.

[11] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In G. B. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, number 1524 in LNCS, chapter 1, pages 9–50. Springer-Verlag, 1998.

[12] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathematics*, 2(2):164–168, 1944.

[13] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[14] H. Park, S. Amari, and K. Fukumizu. Adaptive natural gradient learning algorithms for various stochastic models. *Neural Networks*, 13(7):755–764, 2000.

[15] M. Riedmiller. Advanced supervised learning in multi-layer perceptrons – From backpropagation to adaptive learning algorithms. *Computer Standards and Interfaces*, 16(5):265–278, 1994.

[16] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In E. H. Ruspini, editor, *Proceedings of the IEEE International Conference on Neural Networks*, pages 586–591. IEEE Press, 1993.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error backpropagation. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 318–362. MIT Press, 1986.

[18] M. Toussaint. On model selection and the disability of neural networks to decompose tasks. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2002)*, pages 245–250, 2002.

Institut für Neuroinformatik
Ruhr-Universität Bochum
D-44780 Bochum, Germany
Email address: {christian.igel,marc.toussaint}@neuroinformatik.rub.de

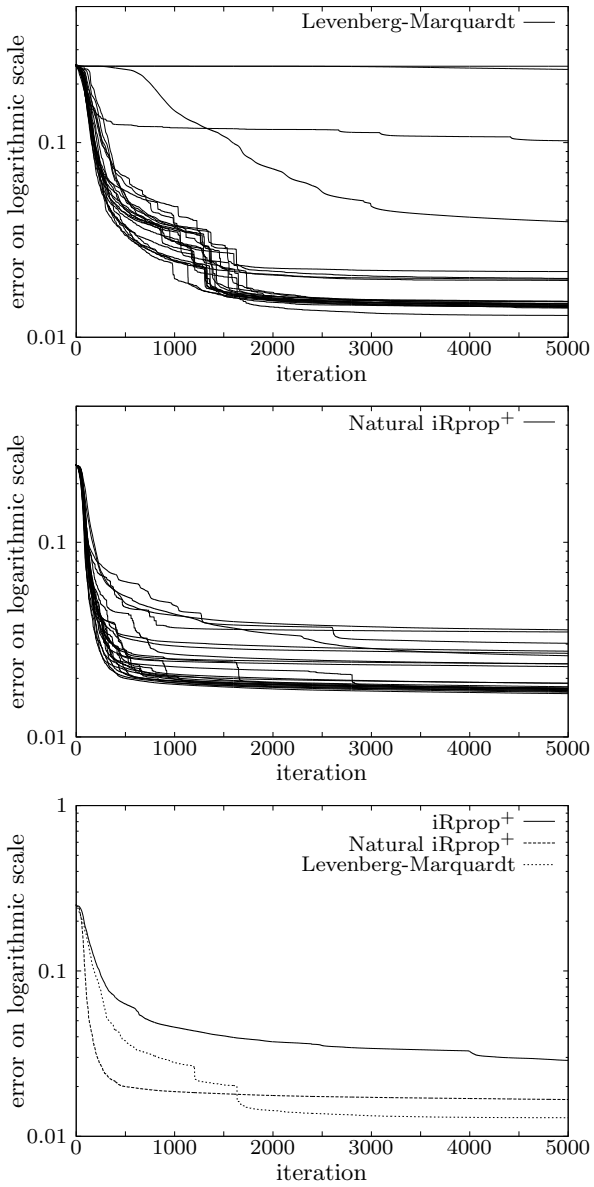

Figure 4: Results for the extended XOR problem. The upper plots show the results for the different settings of λ_0 and λ_- averaged over 20 trials for iRprop$^+$ using natural gradient and Levenberg-Marquardt optimization, respectively. The lower plot shows the averaged trajectories for the parameters resulting in the lowest final error in each case compared to standard iRprop$^+$.

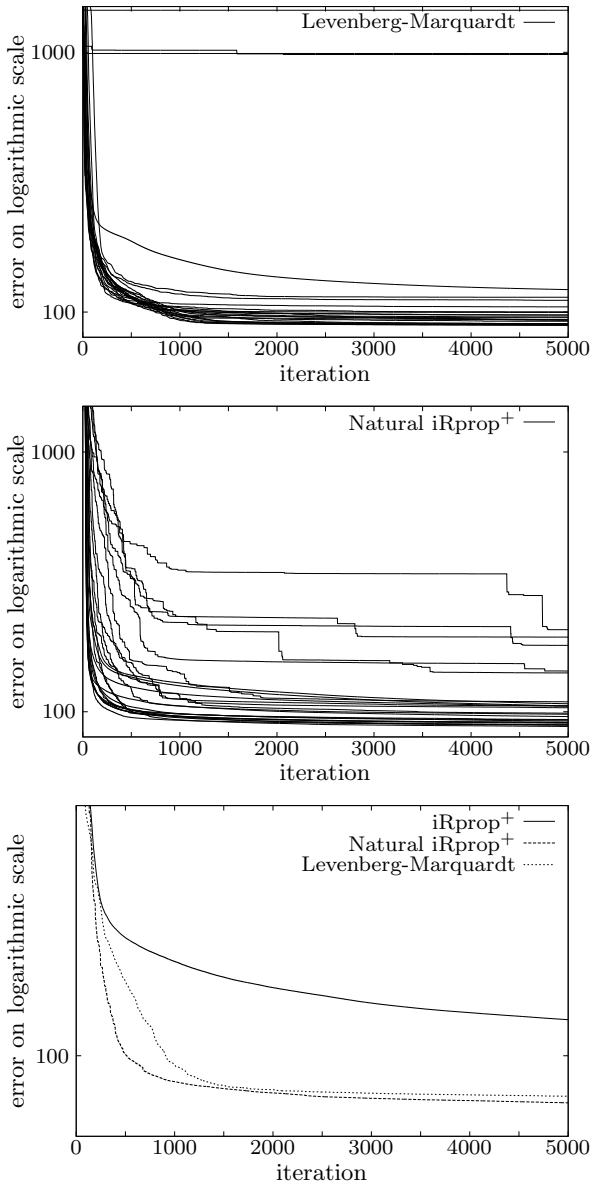

Figure 5: Results for the sunspots prediction problem. The upper plots show the results for the different settings of λ_0 and λ_- averaged over 20 trials for natural iRprop$^+$ and Levenberg-Marquardt optimization, respectively. The lower plot shows the averaged trajectories for the parameters resulting in the lowest final error in each case compared to standard iRprop$^+$.