LU TP 93-13

On Langevin Updating in Multilayer Perceptrons

Thorsteinn Rögnvaldsson[1]

Department of Theoretical Physics, University of Lund,
Sölvegatan 14 A, S-223 62 Lund, Sweden

Submitted to **Neural Computation**

Abstract:
The Langevin updating rule, in which noise is added to the weights during learning, is presented and shown to improve learning on problems with initially ill-conditioned Hessians. This is particularly important for multilayer perceptrons with many hidden layers, that often have ill-conditioned Hessians. In addition, Manhattan updating is shown to have a similar effect.

[1] denni@thep.lu.se

Introduction

There are numerous examples in the literature of cases where performances of artificial neural networks (**ANN**) have been improved by clever use of noise in the training. For instance, their generalization ability can be improved by the use of noise-corrupted training patterns [1, 2, 3, 4, 5] and they can escape local minima with the help of synaptic noise [6, 7, 8]. This paper presents and analyzes such performance improvements achieved with a very simple method, the so-called Langevin updating (**LV**), where gaussian noise is added to the weights during training. This technique has two advantages over other algorithms:

- Initially, the noise helps the network search difficult regions of the error surface.
- Asymptotically, the noise helps the network escape local minima.

Asymptotic properties of LV updating have previously been studied in the field of simulated annealing and results can be transferred directly to ANN learning (see [9] and references therein). Consequently, our focus will instead be on its positive effects on the initial phase of learning. We show that LV updating, due to its ability to explore flat regions on the error surface, often outperforms on-line backpropagation (**BP**) both in convergence speed and quality of the final network solutions. It is also pointed out, and experimentally verified, that Manhattan updating does this too.

The Langevin Updating Rule

The Langevin updating rule with a discrete time step η is given by

$$\Delta\omega(t) = -\eta\nabla E(\omega) + \sqrt{\eta 2T(t)}\xi \tag{1}$$

where $\Delta\omega$ is the weight update, ∇E the gradient of the error function, and ξ a gaussian noise term with unit variance. This equation can also be modified to allow for other noise correlations, for instance if it is desired to vary the noise with the position in weight space [11]. The effect of the Langevin equation is to place the network in a "heat bath" with temperature T and a Gibbs equilibrium distribution. However, since the goal of ANN learning is not to achieve a Gibbs probability distribution we use the simpler form

$$\Delta\omega(t) = -\eta\nabla E(\omega) + \xi(t) \tag{2}$$

where the variance of the gaussian noise ξ is varied during learning.

To make sure the system escapes all local minima and ends up in the global minimum when $t \to \infty$, the temperature in eq. (1) should decrease as

$$T(t) \propto \frac{1}{\log t} \tag{3}$$

for large t [10]. The optimal constant of proportionality, the "critical depth", can be calculated from knowledge about the deepest local minima. This annealing schedule is unfortunately prohibitively slow for practical purposes and we instead use a geometric annealing schedule

$$T(t) = kT(t-1) \tag{4}$$

where $k < 1$. This is selected purely because of its simplicity and works fine here since we study changes during the initial stage of learning. However, a different annealing schedule would be needed for problems where local minima, and not saddle points, are a serious problem.

On-line BP updating, where patterns are drawn at random from the training set and weights are updated after each pattern presentation, also introduces noise to the weights. The noise level depends on the variation of the training patterns, the procedure by which these are chosen, and the location in weight space [12, 13]. In this case, the learning rate η takes the role of the temperature and an annealing procedure similar to eq. (3) is needed to guarantee convergence to the global minima [12]. Furthermore, to achieve optimal convergence, the learning rate for the weights should be tuned with the curvature of the energy surface, which is computationally expensive and practically impossible in most applications.

The method used in this paper is actually an on-line LV, the gradient is evaluated over a subset of the training set before the updating.

Hessian Analysis

ANN learning can be extremely slow if the problem is "stiff" or if the error surface is very flat. The first difficulty, stiffness, is usually handled by using second order training algorithms (see [14] for a review), dynamic learning rates (see [15] and references therein), or simply by proper rescaling of the inputs. The second difficulty, flatness, is on the other hand a serious and common problem in ANN applications [16], that cannot be handled by resorting to second order minimization algorithms. However, LV updating takes care of this problem.

The curvature of the error surface is given by the eigenvalues λ_i of the Hessian matrix $\mathsf{H}\,(\omega) = \nabla^2 E(\omega)$. The absolute ratio of the largest to the smallest eigenvalue is called the *condition number* of H, and H is *ill-conditioned* if the reciprocal of the condition number is very small ("small" meaning close to the floating point precision of the computer). The *nullity* of H is usually defined as the dimension of the weight-subspace spanned by the eigenvectors with eigenvalue zero but is used here as the dimension of the subspace spanned by eigenvectors with $|\lambda_i/\lambda_{max}| < \varepsilon$, where ε is the floating point precision of the computer. An ill-conditioned Hessian with a large "nullity" corresponds to an error surface with very little or no information on the step sizes needed to converge. A normal learning algorithm therefore stalls when it encounters the "nullity". Langevin updating, on the other hand, probes the "nullity" by Brownian motion looking for signs of any curvature. As soon as some curvature is found, the gradient term takes control and forces the motion downhill.

It is instructive to inspect the Hessian for a multilayer perceptron (**MLP**) to understand why small eigenvalues are a general problem. We consider an MLP with one hidden layer with N_h units, N_i inputs, one output unit and the error measure

$$E = \frac{1}{2M}\sum_{p=1}^{M}(o-t)^2 \tag{5}$$

where M is the number of training patterns used in one updating, o the network output and t the target training value. To simplify the Hessian as much as possible and still keep the essential features, we assume that $\sum_p (o-t)/M \approx 0$, which is not uncommon in ANN problems, and that initial weight values are small. This gives an initial Hessian of the general form

$$\mathsf{H} = \begin{pmatrix} c^2 & ..\mathcal{O}(c^3).. & ..\mathcal{O}(c^3\omega).. & ..\mathcal{O}(c^3\omega\langle x\rangle).. \\ . & ..\mathcal{O}(c^4).. & ..\mathcal{O}(c^4\omega).. & ..\mathcal{O}(c^4\omega\langle x\rangle).. \\ . & \ldots & ..\mathcal{O}(c^4\omega^2).. & ..\mathcal{O}(c^4\omega^2\langle x\rangle).. \\ . & \ldots & \ldots & ..\mathcal{O}(c^4\omega^2\langle xx\rangle).. \end{pmatrix} \tag{6}$$

where $\langle\rangle$ denotes averaging over the M training patterns and x is the input signal (indices have been suppressed). The scaling constant c comes from the transfer function and equals 0.5 for a $[0,1]$ sigmoid. With two hidden layers in the MLP we get additional terms of $\mathcal{O}(\omega^3)$, and similarly for more hidden layers. The rightmost column is $N_i \times N_h$ elements wide and the two middle columns are N_h elements wide. The eigenvalue spectrum for a matrix of the form (6) typically has one, or very few, large eigenvalues and several groups of eigenvalues of decreasing magnitudes (see fig. 3a). The sizes of the small eigenvalues decrease with the elements in the rightmost columns. Hence, if the initial weights are small, the risk of having an ill-conditioned Hessian increases with the number of layers in the MLP. Also, since MLP networks tend to have a "fan-in" structure, i.e. more input units than output units, the subspace corresponding to small eigenvalues (the "nullity") will be larger than the subspace associated with large eigenvalues.

Note on the Manhattan algorithm

As a digression, we note that random search of the "nullity" is also achieved with the Manhattan (**MH**) updating rule [17]:

$$\Delta\omega(t) = -\eta_M \ \mathrm{sign}\left[\nabla E(t)\right] \tag{7}$$

which could explain why the MH algorithm has been more successful than BP on some problems [17, 23].

To see this we rewrite eq. (7) as

$$\Delta\omega(t) = -\eta_M \ \mathrm{sign}\left[\nabla E(t)\right] = -\eta\nabla E(t) + \delta\omega(t) \tag{8}$$

and study the distribution of the "corrections" $\delta\omega$. The gradient $\eta\nabla E(t)$ can be considered a random variable for weights inside the "nullity". Denote the distribution of $\eta\nabla E$ at time t by $P[-\eta\nabla E(t)]$ and the distribution of corresponding corrections $\delta\omega$ by $\rho[\delta\omega(t)]$. If the same learning rate is used for all the weights inside the "nullity", the MH updating rule collapses $P[-\eta\nabla E(t)]$ into two spikes located at $\pm\eta_M$, resulting in the following relation between P and ρ:

$$\rho[\delta\omega] = \begin{cases} P[-\eta_M - \delta\omega] & \text{if } \delta\omega > \eta_M \\ P[\eta_M - \delta\omega] + P[-\eta_M - \delta\omega] & \text{if } |\delta\omega| < \eta_M \\ P[\eta_M - \delta\omega] & \text{if } \delta\omega < -\eta_M \end{cases} \tag{9}$$

The distribution $P[-\eta\nabla E(t)]$ is usually symmetric and centered around $\nabla E(t) = 0$. Assuming a gaussian shape for $P[-\eta\nabla E(t)]$ with a width of σ we get two probable scenarios: For $\eta_M > \sigma$, $\rho[\delta\omega]$ will be composed of two peaks at $\pm\eta_M$, and for $\eta_M < \sigma$, $\rho[\delta\omega]$ looks like a truncated gaussian. The first scenario corresponds to a random walk with fixed step lengths (as expected), whereas the latter boils down to LV updating. Both result in a random search of the "nullity".

Monte Carlo Experiments

Methods

Four different problems are used to test LV and MH versus BP updating. They are chosen as representatives of different problem areas in ANN applications; classification with real-valued inputs and binary inputs, prediction, and one real world classification task. The performance is measured on an ensemble of 100 networks to reduce effects of statistical fluctuations. Eq. (2) is used for LV and BP, with zero noise in the latter case, and eq. (7) is used for MH updating, and a momentum term is added in all three cases. We use a "block" updating scheme where the error gradient is summed over a randomly chosen subset, typically 10 patterns, of the training set before each weight updating. A mean square error, eq. (5), is used so that the learning rate η is independent of the block size. Before training, a coarse search is done to find optimal learning parameters. During training, η is adjusted with a simple "bold driver" technique; we increase η geometrically if the error is decreasing, decrease it otherwise. In the MH case, the learning rate η_M is monotonically decreased geometrically. The LV noise level is initially set to 0.01 for all the problems and is decreased geometrically to practically zero at the end of training. These heuristic parameter adjustment schemes are all chosen for their simplicity, not for optimality.

All simulations are done with the feed-forward network simulation package `JETNET 2.2` [18].

Results

The first problem is to separate two **overlapping 10-dimensional gaussian distributions** with the same mean but different standard deviations. Earlier studies indicate that LV or MH updating is essential to achieve high quality results on this problem [17, 19]. Networks with 10 inputs, 20 hidden and one output unit are trained for 100 epochs, with 1000 pattern presentations per epoch. Initial weight values are $\omega \in [-0.01, 0.01]$ and momentum is set to $\alpha = 0.5$. Figure 1a shows the classification performances for different values of the learning rate. The LV solutions are better and have a smaller variance than the BP ones.

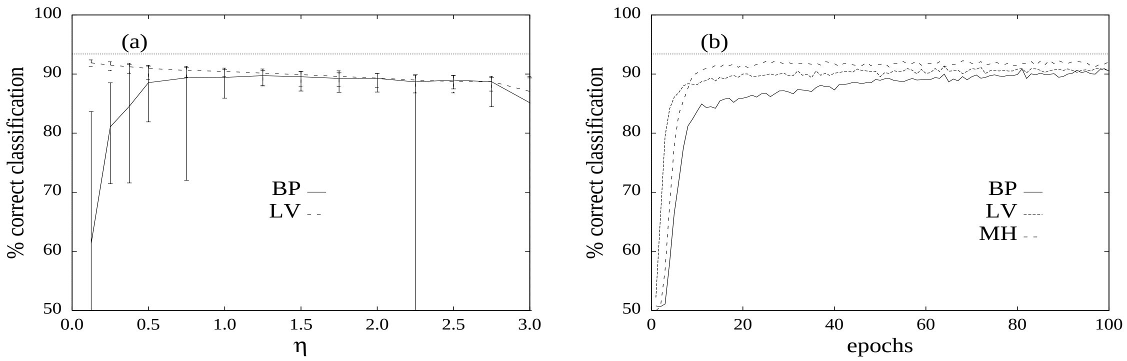

Figure 1: Classification performances on the overlapping gaussian problem. **(a)** Average, best and worst final performances using BP and LV for different values of the learning rate η. **(b)** Average classification performance during training using MH with $\eta_M = 0.005$, LV with $\xi = 0.01$ and $\eta = 1$, and BP with $\eta = 1$. The line at 93.4% indicates the maximum classification possible on this problem.

The second problem is to predict **the Mackey-Glass time series** at time $x(t+85)$ given $x(t), x(t-6), x(t-12)$ and $x(t-18)$. This problem has been used in several benchmark tests on learning the dynamics of time series (see [20] and references therein). The data is translated to its mean value, which speeds up learning. The

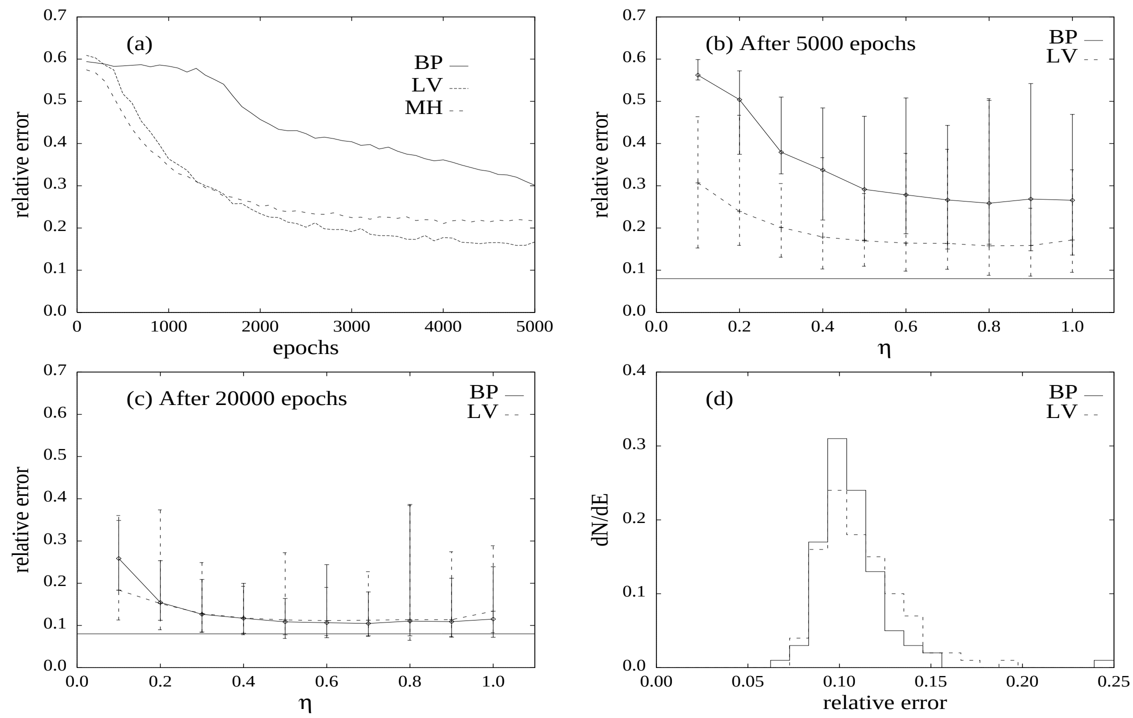

Figure 2: Relative error (rms error/rms deviation of data) for the Mackey-Glass prediction task: **(a)** Average values during the first 5000 epochs for $\eta = 0.7$ and $\eta_M = 0.007$.**(b)** Average, best and worst relative error for BP and LV updating, after 5000 epochs, for different values of the learning rate η. **(c)** The same after 20000 epochs.**(d)** Distribution of final errors after 20000 epochs for $\eta = 0.6$. The solid line in (a) and (b) marks the lowest error reported in ref. [20].

networks have 4 inputs, two hidden layers with 10 and 5 sigmoidal units, and one linear output unit. The training and test set contain 500 patterns each and the networks are trained for 20000 epochs. The weights are initialized with $\omega \in [-0.01, 0.01]$ and momentum is $\alpha = 0.3$. Adding noise, by means of LV or MH, significantly shortens the convergence time, although η_M is tricky to tune and MH therefore deteriorates later on (fig. 2a). After 20000 epochs BP catches up with LV as the errors reach their lower limit (fig. 2c). By this time the noise level in LV is zero and there is no difference between LV and BP.

The third problem, **n-parity**, has previously been used in benchmark studies where Conjugate Gradient algorithms greatly outperformed BP [21]. We study 4, 5 and 6 dimensional parity problems, with the architecture n inputs, 8 hidden units and one output. Using more hidden units than inputs decreases the risk of getting stuck in a local minima. Each network is trained for 10000 epochs or until 100% correct classification. The weights are initialized with $\omega \in [-0.1, 0.1]$. A large momentum term is used with $\alpha = 0.9$. The results are listed in table 1. Adding noise does not improve the convergence in any significant way and the MH performance is outright miserable.

	4-Parity			5-Parity			6-Parity		
Upd.	η	$\langle\tau\rangle$	conv.	η	$\langle\tau\rangle$	conv.	η	$\langle\tau\rangle$	conv.
BP	0.5	1088	99%	0.5	2202	83%	0.5	4111	49%
MH	0.05	393	42%	0.07	390	4%	0.005	-	0%
LV	0.5	1034	97%	0.5	2196	87%	0.5	3396	48%

Table 1: Average convergence times τ for those networks that learned the parity problem within 10000 epochs. The rightmost column of each section shows the percentage of 100 networks that managed to converge.

Thyroid classification with MLP						
	Training set		Generalization			
Updating	$\langle E\rangle$	$E_{\min}$	$\langle E\rangle$	$E_{\min}$	$\langle$Class.$\rangle$	Class.$_{\text{best}}$
BP, $\eta = 20.0$	11.5	2.0	108.8	88.0	98.2%	98.6%
LV, $\eta = 20.0$	15.6	5.4	110.7	89.5	98.2%	98.7%

Table 2: Performance of MLP on classification of hypothyroid functioning. Angle brackets denote averages. The rightmost column shows the best classification performance achieved. The latter should be compared with 98.5%, which is the best classification performance reported in ref. [15].

The fourth problem is to **determine whether a patient is hypothyroid or not** [22]. It is a real world data set that has recently been used as "a very hard practical classification task" in a benchmark test of different ANN training algorithms [15], with the conclusion that a variation of MH updating with dynamic individual learning rates is the quickest learning algorithm. The inputs are normalized to zero mean and a root mean square deviation of unity. This normalization shortens the convergence time with an order of magnitude compared to the results in [15]. The same network architecture and weight initialization procedure is used as in [15], and the momentum term is set to zero. Table 2 shows the average training and generalization errors for networks trained 1000 epochs each. No improvement is made with LV updating (MH was not tested on this problem).

Discussion

General Results

The initial Hessian (i.e. evaluated in the initial weight configuration) eigenvalue spectra in fig. 3, together with the experimental results, verify that LV and MH updating

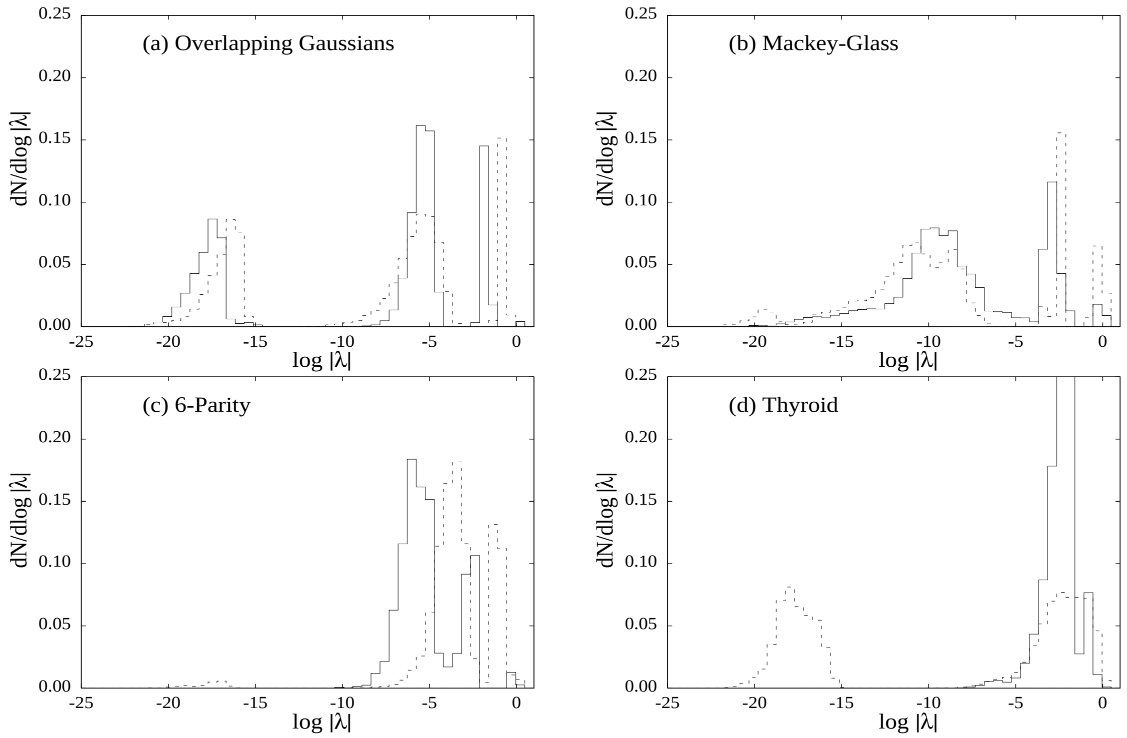

Figure 3: Initial Hessian eigenvalue spectra for an ensemble of 100 networks for **(a)** the overlapping gaussian distributions, **(b)** the Mackey-Glass time series, **(c)** the six-dimensional parity problem, and **(d)** the thyroid classification problem. The solid line shows the eigenvalue spectrum evaluated over the complete training set and the dashed line shows the spectrum evaluated over randomly chosen subsets of patterns. All eigenvalues are computed with double precision.

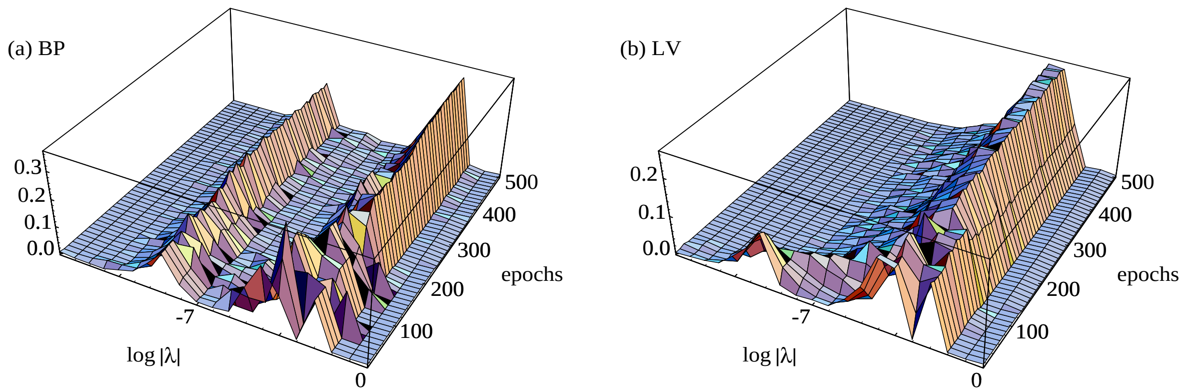

Figure 4: Time evolution of the eigenvalue spectra for the overlapping gaussian problem (one network) when using **(a)** backpropagation and **(b)** Langevin updating. Eigenvalues are computed with single precision.

outperforms BP on problems with an initial flatness problem. Both problems where LV updating helps have Hessians that could be termed "ill-conditioned", whereas the other two problems do not. Furthermore, the dynamics of the eigenvalue spectrum, shown in fig. 4, strikingly demonstrates the difference between BP and LV updating. The LV noise immediately helps the network find regions with larger curvature, while on-line BP has a very hard time leaving the "nullity", if it ever leaves it! The LV training is also more robust than BP and less sensitive to the specific values of the training parameters, as can be seen from figs.1 and 2. The average performance of LV is less sensitive than the BP one to different choices of learning rates. The results are also quite insensitive to the initial noise level (the same initial noise level was used in all four problems) and the final result is the same for noise levels within two orders of magnitude, downwards, from the one used here.

When evaluating the condition of the Hessian H, it is of course the eigenvalue spectrum determined for the whole training set that is important. Different data subsets may produce different "subnullities", but if these "subnullities" have no overlap, the effect on the training will be negligible (see fig. 3d).

We have verified these results on a fifth problem; predicting the solar insolation onto a building using a two hidden layer MLP [24]. The initial Hessian for this problem has a condition number of $\mathcal{O}(10^{15})$ and LV updating outperforms all other attempts on the problem.

Previous Results

At a first glance, these results seem to contradict previous studies where learning of the parity problem is improved with synaptic noise [6, 7]. The parity problem is not ill-conditioned and should consequently not be easier to learn for LV updating. There is, however, no disagreement. In both these studies the effect comes from escaping local minima that are encountered late in the training; in [7] by keeping a constantly high noise level (20%), and in [6] by increasing the noise level if the weight makes mistakes. This effect is not present in the results reported here, the noise is decreased too fast to aid escape from local minima encountered late in the learning process.

Conjugate Gradients

We have also tested the Conjugate Gradient algorithms on the overlapping gaussians and the Mackey-Glass time series, but they never reach final performances compatible with BP or LV. Similar results on the thyroid classification problem are reported in [15].

Acknowledgements

I want to thank Tom Heskes and two anonymous reviewers for useful pointers to references. This research was sponsored in part by the Göran Gustafsson Foundation for Research in Natural Sciences and Medicine.

References

[1] Yau, H., and Wallace, D. 1991. Enlarging the Attractor Basins of Neural Networks with Noisy External Fields. *J. Phys. A* **24**, 5639-5650.

[2] Sietsma, J., and Dow, R. 1991. Creating Artificial Neural Networks That Generalize. *Neur. Netw.* **4**, 67-79

[3] Krogh, A., and Hertz, J. 1992. Generalization in a Linear Perceptron in the Presence of Noise. *J. Phys. A* **25**, 1135-1147.

[4] Holmström, L., and Koistinen, P. 1992. Using Additive Noise in Back-Propagation Training. *IEEE Trans. Neur. Netw.* **3**, 24-38

[5] Bishop, C. 1993. Training with Noise is Equivalent to Tikhonov Regularization. Subm. to *Neur. Comp.*.

[6] Hanson, S. 1990. A Stochastic Version of the Delta Rule. *Physica* **D 42**, 265-272.

[7] Murray, A. 1992. Multilayer Perceptron Learning Optimized for On-Chip Implementation: A Noise-Robust System. *Neur. Comp.* **4**, 366-381.

[8] Murray, A., and Edwards, P. 1993. Synaptic Weight Noise During MLP Learning Enhances Fault-Tolerance, Generalisation and Learning Trajectory. *Adv. in Neur. Inf. Proc. Syst.* **5**, Hanson, Cowan, and Giles (eds.), 491-498. Morgan Kaufmann, San Mateo, CA.

[9] Guillerm, T., and Cotter, N. 1991. A Diffusion Process for Global Optimization in Neural Networks. *Proc. Int. Joint Conf. on Neur. Netw. 1991*, IEEE Press.

[10] Kushner, H. 1987. Asymptotic Global Behavior for Stochastic Approximation and Diffusion with Slowly Decreasing Noise Effects: Global Minimization via Monte Carlo. *SIAM J. Appl. Math.* **47**, 169-185.

[11] Söderberg, B. 1988. On the Complex Langevin Equation. *Nucl. Phys.* **B295**, 396-408.

[12] Heskes, T., Slijpen, E., and Kappen, B. 1993. Cooling Schedules for Learning in Neural Networks. *Phys. Rev.* **E 47**, 4457-4464.

[13] Leen, T., and Moody, J. 1993. Weight Space Probability Densities in Stochastic Learning: I. Dynamics and Equilibria. *Adv. in Neur. Inf. Proc. Syst.* **5**, Hanson, Cowan, and Giles (eds.), 451-458. Morgan Kaufmann, San Mateo, CA.

[14] Battiti, R. 1992. First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method. *Neur. Comp.* **4**, 141-166.

[15] Schiffmann, W., Joost, M., and Werner, R. 1993. Comparison of Optimized Backpropagation Algorithms. *Proc. ESANN'93*, Brussels.

[16] Saarinen, S., Bramley, R., and Cybenko, G. 1993. Ill-conditioning in Neural Network Training Problems. *SIAM J. Sci. Comp.* **14**, 693-714.

[17] Peterson, C. and Hartman, E. 1989. Explorations of the Mean Field Theory Learning Algorithm. *Neur. Netw.* **2**, 475-494.

[18] Lönnblad, L., Peterson, C., and Rögnvaldsson, T., work in progress.

[19] Rögnvaldsson, T. 1993. Pattern Discrimination using Feed-forward Networks. *Neur. Comp.* **5**, 483-491.

[20] Hartman, E., and Keeler, J. 1991. Predicting the Future: Advantages of Semilocal Units. *Neur. Comp.* **3**, 566-578.

[21] Johansson, E., Dowla, F., and Goodman, D. 1992. Backpropagation Learning for Multilayer Feed-forward Neural Networks using the Conjugate Gradient Algorithm. *Int. J. Neur. Syst.* **2**, 291-301.

[22] Quinlan, J. 1987. Simplifying Decision Trees. *Int. J. Man-Machine Stud.*, 221.

[23] Lönnblad, L., Peterson, C., and Rögnvaldsson, T. 1992. Mass Reconstruction with a Neural Network. *Phys. Lett.* **B** 278, 181-186.

[24] Ohlsson, M., Peterson, C., Pi, H., Rögnvaldsson, T., and Söderberg, B. 1994. "Predicting Utility Loads with Artificial Neural Networks – Methods and Results from the Great Energy Predictor Shootout". To appear in *1994 ASHRAE Trans.* **100**, part 2.