LU TP 93-13

June 1993

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 analyzed. It is well controlled and, being a natural extension to standard backpropagation learning, easily combined with other modifications of backpropagation. If the Hessian matrix is numerically ill-conditioned, Langevin updating converges faster than backpropagation and, probably, also higher order algorithms. This is particularly important for multilayer perceptrons with many hidden layers, which tend to have ill-conditioned Hessians. In addition, Manhattan updating is shown to have a similar effect as Langevin updating.

[1] denni@thep.lu.se

Introduction

Performances of artificial neural networks (**ANN**) are often improved when external noise is present during the training phase. For instance in Hopfield-type networks the basins of attraction for the stored memory patterns are enlarged when noise-corrupted training patterns are used [1]. In linear perceptrons the generalization ability can be improved with noisy training patterns [2] and in hardware implementations of multilayer perceptrons both generalization and learning abilities are improved if noise is introduced into the system (see [3] and references therein).

This paper presents and analyzes the so-called Langevin updating, where noise is added directly to the weights during training. It is a natural and benevolent extension to backpropagation (**BP**), easily combined with other variations of BP, with modest additional computational needs. Langevin updating often outperforms BP both in convergence speed and in quality of the final solutions on problems where higher order methods are not beneficial. This improved convergence is shown below to be associated with an ill-conditioned Hessian matrix. It is also discussed that Manhattan updating should work well on ill-conditioned problems due to its similarity with Langevin updating.

The Langevin Updating Rule

The Langevin (**LV**) updating rule reads

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

where $\Delta\omega$ is the weight update, η the learning rate, ∇E the gradient (with respect to ω) of the error function, and ξ a Gaussian noise term.

The virtues of LV updating are:

- It places the network in a "heat bath" generating weight configurations within the Boltzmann ensemble. This allows occasional steps with increasing error, whereby the network escapes local minima. The noise level ξ reflects the "temperature" of the heat bath and is decreased as training proceeds, annealing the network into a stable final solution.
- The noise is independent of the training patterns. This avoids the so-called "flat-spot" problem when the nodes are saturated and the derivative, which is propagated backwards, becomes approximately zero [4].

- Variations of standard BP, like extra complexity terms, alternative error measures, etc. can be combined with LV updating without any modifications.
- The noise level affecting the weights is easily controlled

The latter is not the case when noise is inserted into the nodes of the network or via stochastic BP updating. When noise is added to the nodes, input or others, the noise affecting successive layers in a network depends on the weights. In stochastic BP noise is introduced by updating over randomly chosen subsets of training patterns. The noise level hence depends both on the variation of the training patterns and the procedure by which these are chosen for updating. Maximum noise level is achieved when the weights are updated after each pattern presentation. Such an updating procedure is time consuming and it is preferable to use a block update scheme with LV updating instead and decrease the total number of operations. Also, though stochastic BP escapes local minima [5], it does not automatically avoid the "flat spot" problem.

Hessian Analysis

The reason LV updating can outperform other algorithms lies in the fact that these tend to be extremely slow under certain circumstances. This can for instance happen if the problem is "stiff" (i.e. the weights grow with different time scales) or if the error surface is very flat. Stiffness can often be avoided if inputs are properly normalized but is also well handled by second order training algorithms (see [6] for a review) or individual dynamic learning rates (see [7, 8, 9] and references therein). Flat error surfaces, on the other hand, are unfortunately both common in ANN applications and a severe problem for first and second order minimization methods [10].

The curvature of the error surface is given by the eigenvalues λ_i of the Hessian matrix $\mathbf{H} = \nabla^2 E$. It is instructive to inspect the Hessian for a multilayer perceptron (**MLP**) to understand why small eigenvalues are a general problem. Consider an MLP with one hidden layer with N_h units, N_i inputs, one output unit and an error measure

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

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

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

where a bar means 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 *condition number* of $\mathbf{H}$ is defined as the absolute ratio of the largest to the smallest eigenvalue and $\mathbf{H}$ is *ill-conditioned* if the reciprocal of the condition number is close to the floating point precision of the computer [11]. In the case of single precision floating point operations this is approximately 10^{-6}. The *nullity* of $\mathbf{H}$ is strictly defined as the dimension of the weight-subspace spanned by the eigenvectors with eigenvalue zero [11] but is used here as the dimension of the subspace spanned by eigenvectors with $|\lambda_i/\lambda_{max}| < 10^{-6}$. An ill-conditioned Hessian with a large "nullity" contains very little information on the step sizes and direction needed to converge. A normal learning algorithm therefore stalls when it encounters the "nullity". LV updating, however, probes the "nullity" by Brownian motion looking for signs of any curvature. As soon as some curvature is found, the gradient term forces the motion downhill. The eigenvalue spectrum for a matrix of the form (3) typically has one, or very few, large eigenvalues and several groups of small eigenvalues (see fig. 3a). The magnitudes of the small eigenvalues decrease with the sizes of 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.

Relation to the Manhattan Algorithm

Random search of the "nullity" can also be achieved with the Manhattan (**MH**) updating rule [12]:

$$\Delta\omega(t) = -\eta_M \ \mathrm{sign}\left[\nabla E(t)\right] \tag{4}$$

To see this we rewrite eq. (4) as

$$\Delta\omega(t) = -\eta_M \ \mathrm{sign}\left[\nabla E(t)\right] = -\eta\nabla E(t) + \delta\omega(t) \tag{5}$$

and study the distribution of the "corrections" $\delta\omega$. If a block update scheme is used, $\eta\nabla E(t)$ 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 the weights, the MH updating rule collapses $P[-\eta\nabla E(t)]$ into two spikes located at $\pm\eta_M$, which gives the following relation between P and ρ:

$$\rho[\delta\omega] = \begin{cases} P[-\eta_M - \delta\omega] & \text{if } \delta\omega > \eta_M \\ P[\eta_M - \delta\omega] + P[-\eta_M - \delta\omega] & \text{if } |\delta\omega| < \eta_M \\ P[\eta_M - \delta\omega] & \text{if } \delta\omega < -\eta_M \end{cases} \tag{6}$$

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, whereas the latter boils down to LV updating. Both result in a random search of the "nullity".

Numerical Tests

The LV and MH algorithms are compared with the BP algorithm on four problems. The performance is measured on an ensemble of 100 networks to reduce effects of statistical fluctuations. A block updating scheme is used for all algorithms, where the error gradient is summed over a randomly chosen subset, typically $\mathcal{O}(10)$ patterns, of the training set before each weight updating. A normalized summed square error, eq. (2), is used so that the learning rate η is independent of the number of patterns. Before training, a coarse search is done to find optimal learning parameters. During training, the MH learning rate η_M is decreased geometrically, whereas η is adjusted with a simple "bold driver" technique; increase η if the error is decreasing, decrease it otherwise. The LV noise level is decreased geometrically to practically zero at the end of training. All simulations are done with `JETNET 2.2` [13].

The first problem is to separate two **overlapping 10-dimensional Gaussian distributions** with the same mean value but different standard deviations. Earlier studies indicate that LV or MH updating is essential to achieve high quality results on this problem [12, 14]. Networks with 10 inputs, 20 hidden and one output unit are trained for 100 epochs, with 1,000 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. With initial noise $\xi = 0.01$ the LV solutions are generally better and have a smaller variance than the BP ones (results are similar for $\xi \in [10^{-5}, 10^{-2}]$). LV is also more stable for different learning rates and needs less parameter fine-tuning. The convergence curves in fig. 1b show that noisy updating, in the form of LV or MH, immediately reaches

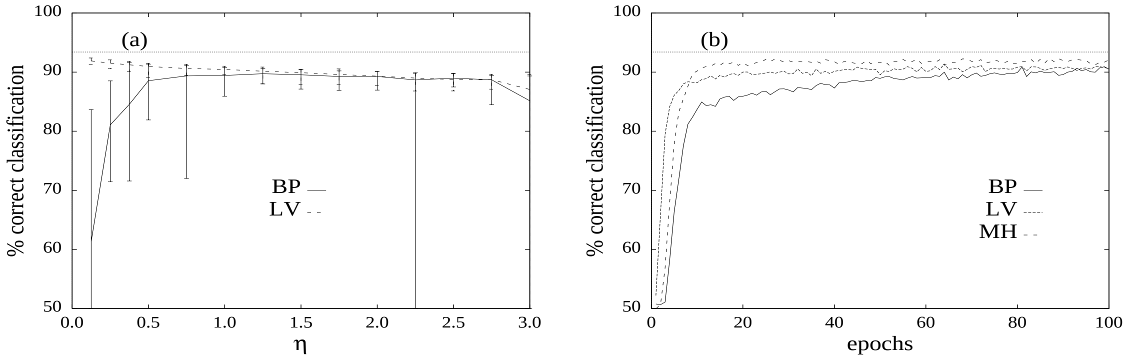

Figure 1: Classification performances on the overlapping Gaussian problem. **(a)** Average, best and worst final performances using BP and LV for different values of the learning rate η. **(b)** Average classification performance using MH with $\eta_M = 0.005$, LV with $\xi = 0.01$ and $\eta = 1$, and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

better classification performances than BP.

The second problem is to predict **the Mackey-Glass time series** [15] 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 [16] and references therein). The time series is translated to its mean value, which speeds up learning. The networks use 4 inputs, two hidden layers with 10 and 5 sigmoidal units respectively, and one linear output unit. The training and test set contain 500 patterns each and the networks are trained for 20,000 epochs. The weights are initialized with $\omega \in [-0.01, 0.01]$ and momentum is $\alpha = 0.3$. The LV updating reaches a smaller average relative error with a smaller variance than BP (fig. 2a). 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. 2b).

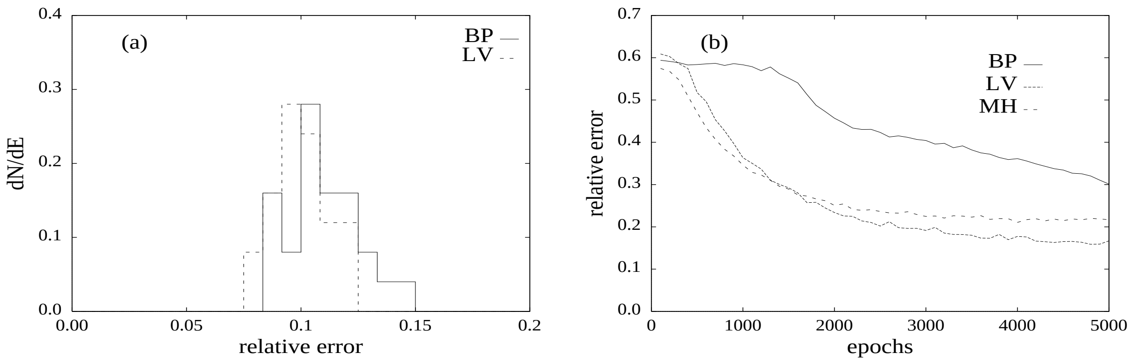

Figure 2: **(a)** Final distribution of relative errors after training for 20,000 epochs on the Mackey-Glass time series. The learning rate is $\eta = 0.7$ for both BP and LV. The average relative errors are $\overline{E} = 0.110$ and $\overline{E} = 0.100$ for BP and LV respectively. **(b)** Average convergence curves for the first 5,000 epochs. The MH learning rate is $\eta_M = 0.007$.

The third problem, **parity** [18], has been used in benchmark studies where Conjugate Gradient (**CG**) algorithms greatly outperformed BP [17]. I use 4, 5 and 6 dimensional parity problems, with architectures of n inputs, 8 hidden units and one output. Each network is trained for 10,000 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 initial LV noise level is set to $\xi = 0.01$ – a higher noise level slows down the convergence, a lower makes no difference. Table 1 lists the results. 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 for those networks that learned the parity problem within 10,000 epochs. The rightmost column of each section shows the percentage of 100 networks that managed to converge.

The fourth problem is to **determine whether a patient is hypothyroid or not** [19]. 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 [7], with the conclusion that a variation of MH updating with dynamic individual learning rates [20] is the quickest learning algorithm. I normalize the input channels to zero mean and a standard deviation of unity[2]. I use the same network architecture and weight initialization as in [7], zero momentum and a noise level of $\xi = 0.01$. Table 2 shows the average training and generalization errors for networks trained $1,000$ epochs each. No improvement is made with LV updating (MH was not tested on this problem).

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.

These results, together with the Hessian eigenvalue spectra in fig. 3, confirm the statement made in the analysis section. The first two problems, where LV updating

[2] This normalization shortens the convergence time with an order of magnitude, compared to the results reported in [7]

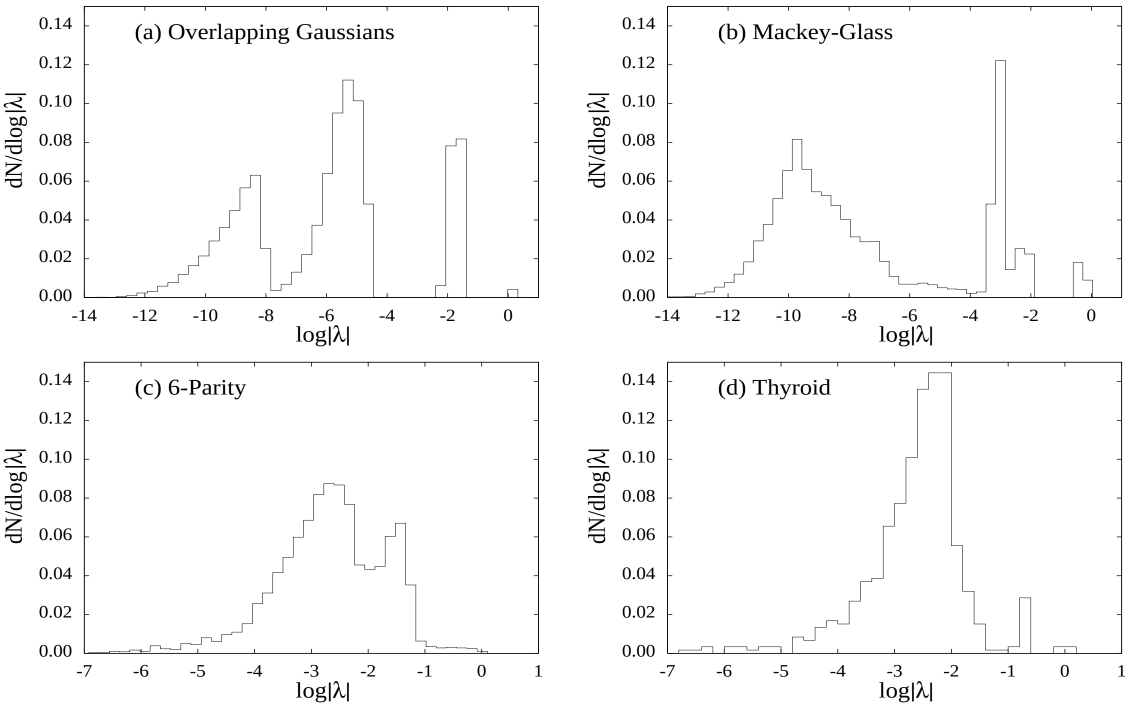

Figure 3: Initial Hessian eigenvalue spectra for an ensemble of 100 networks for **(a)** the overlapping Gaussian distributions, **(b)** the Mackey-Glass time series, **(c)** the six-dimensional parity problem, and **(d)** the thyroid classification problem.

improved the performance, have ill-conditioned Hessians with fairly large "nullities". The last two problems, where LV updating made no difference, have well-conditioned Hessians. Conjugate Gradient algorithms were also tried on the overlapping Gaussians and the Mackey-Glass time series but they converge slowly and do not reach final performances compatible with BP or LV. This also applies to the problem with the thyroid classification [7]. This verifies that LV is indeed a better alternative than other algorithms on problems with ill-conditioned Hessian matrices.

Summary

The Langevin updating rule was presented and demonstrated to be more efficient than deterministic backpropagation updating on problems with ill-conditioned Hessians and large "nullities". Since ill-conditioned Hessians present severe difficulties to second order and conjugate gradient algorithms, Langevin updating will be more efficient than these as well. The addition of the noise term in Langevin updating results in less parameter tuning and higher quality of the final solutions. It is also simple, easy to control, completely compatible with other extensions of backpropagation, and requires very little extra computational effort. Virtues that, in my view, make it a default rather than an optional extension to standard backpropagation.

Also, The Manhattan updating rule was shown to be similar to the Langevin one explaining some improvements observed in the literature with the former [12, 21].

Acknowledgements

This research was sponsored in part by the Göran Gustafsson Foundation for Research in Natural Sciences and Medicine.

References

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

[2] A. Krogh "Learning with Noise in a Linear Perceptron", *J. Phys. A* **25**, 1119 (1992);
A. Krogh and J. Hertz, "Generalization in a Linear Perceptron in the Presence of Noise", *J. Phys. A* **25**, 1135 (1992)

[3] A. Murray, "Multilayer Perceptron Learning Optimized for On-Chip Implementation: A Noise-Robust System", *Neur. Comp.* **4**, 366 (1992);
A. Murray, "Analogue Neural VLSI: Issues, Trends and Pulses", in I. Aleksander and J. Taylor (eds.) *Artificial Neural Networks - Proc. of ICANN 92*, Elsevier, Amsterdam (1992)

[4] S. Fahlman, " An Empirical Study of Learning Speed in Back-Propagation Networks", Carnegie-Mellon University Report CMU-CS-88-162;
S. Fahlman "Faster-Learning Variations on Back-Propagation: An Empirical Study" in *Proc. 1988 Conn. Models Summer School*, D. Touretzky, G. Hinton, and T. Sejnowski (eds.), Morgan Kaufmann, Los Altos CA (1988)

[5] T. Heskes, E. Slijpen and B. Kappen, "Learning in Neural Networks with Local Minima", *Phys. Rev. A* **46**, 5221 (1992)

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

[7] W. Schiffmann, M. Joost and R. Werner, "Comparison of optimized backpropagation algorithms", *Proc. ESANN'93*, Brussels (1993)

[8] T. Heskes and B. Kappen, "Learning-parameter Adjustment in Neural Networks", *Phys. Rev. A* **45**, 8885 (1992)

[9] C. Darken, J. Chang and J. Moody, "Learning Rate Schedules for Faster Stochastic Gradient Search", *Neur. Netw. for Signal Processing 2 - Proc. of the 1992 IEEE Workshop*, IEEE Press, Piscataway NJ (1992)

[10] S. Saarinen, R. Bramley and G. Cybenko, "The Numerical Solution of Neural Network Training Problems", *SIAM J. on Sci. and Stat. Comp.*, April (1993)

[11] W. Press, S. Teukolsky, W. Vetterling and B. Flannery, *Numerical Recipes in C*, Cambridge Univ. Press, Cambridge (1992)

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

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

[14] T. Rögnvaldsson, "Pattern Discrimination using Feed-forward Networks", *Neur. Comp.* **5**, 483 (1993)

[15] M. Mackey and L. Glass, "Oscillations and Chaos in Physiological Control Systems", *Science* **197**, 287 (1977)

[16] E. Hartman and J. Keeler, "Predicting the Future: Advantages of Semilocal Units", *Neur. Comp.* **3**, 566 (1991)

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

[18] D. Rumelhart and J. McLelland (eds.) *Parallel Distributed Processing* (Vol. 1), MIT Press (1986)

[19] J. Quinlan, "Simplifying decision trees", *Int. J. Man-Machine Studies*, 221 (1987)

[20] M. Riedmiller and H. Braun, "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm", in H. Ruspini (ed.) *Proc. of ICNN*, San Fransisco (1993)

[21] L. Lönnblad, C. Peterson, and T. Rögnvaldsson, "Mass Reconstruction with a Neural Network", *Phys. Lett.* **B** 278, 181 (1992)