Technical Report IDSIA-09-99

Local Gain Adaptation in Stochastic Gradient Descent*

Nicol N. Schraudolph
nic@idsia.ch

IDSIA, Corso Elvezia 36
6900 Lugano, Switzerland
http://www.idsia.ch/

March 8, 1999
revised May 15, 1999

Abstract

Gain adaptation algorithms for neural networks typically adjust learning rates by monitoring the correlation between successive gradients. Here we discuss the limitations of this approach, and develop an alternative by extending Sutton's work on linear systems to the general, nonlinear case. The resulting online algorithms are computationally little more expensive than other acceleration techniques, do not assume statistical independence between successive training patterns, and do not require an arbitrary smoothing parameter. In our benchmark experiments, they consistently outperform other acceleration methods, and show remarkable robustness when faced with non-i.i.d. sampling of the input space.

1 Introduction

In recent years, many advanced optimization techniques have been developed or adapted for neural network training. In situations where online learning is required or preferable, however, the vast majority of these techniques are not applicable, and stochastic gradient descent remains the algorithm of choice. The central problem here is how to set the local learning rate, or *gain* of the algorithm, for rapid convergence. *Normalization* methods [2, 3, 4] *calculate* the optimal gain under simplifying assumptions — which may or may not model a given situation well. Even such "optimal" algorithms as Kalman filtering can thus be outperformed by *adaptation* methods which *measure* the effects of finite step sizes in order to incrementally adapt the gain [5].

Unfortunately most of the existing gain adaptation algorithms for neural networks adapt only a single, global learning rate [6, 7], can be used only in batch training [8, 9, 10, 11, 12, 13], or both [14, 15, 16, 17]. Given the well-known advantages of stochastic gradient descent, it would be desirable to have online methods for local gain adaptation in nonlinear neural networks. The first such algorithms have recently been proposed [18, 19]; here we develop a more sophisticated alternative by extending Sutton's work on linear systems [5, 20] to the general, nonlinear case. The resulting *stochastic meta-descent* (SMD) algorithms support online learning with unprecedented speed and robustness of convergence.

*Reprinted from [1].

2 The SMD Algorithm

Given a sequence $\vec{x}_0, \vec{x}_1, \ldots$ of data points, we minimize the expected value of a twice-differentiable loss function $f_{\vec{w}}(\vec{x})$ with respect to its parameters $\vec{w}$ by stochastic gradient descent:

$$\begin{aligned} \vec{w}_{t+1} &= \vec{w}_t + \vec{p}_t \, \vec{\delta}_t \,, \qquad (1) \\ \text{where} \quad \vec{\delta}_t &\equiv -\frac{\partial f_{\vec{w}_t}(\vec{x}_t)}{\partial \vec{w}} \end{aligned}$$

The local learning rates $\vec{p}$ are best adapted by exponentiated gradient descent [21, 22], so that they can cover a wide dynamic range while staying strictly positive:

$$\begin{aligned} \ln \vec{p}_t &= \ln \vec{p}_{t-1} - \mu \frac{\partial f_{\vec{w}_t}(\vec{x}_t)}{\partial \ln \vec{p}} \\ \vec{p}_t &= \vec{p}_{t-1} \exp(\mu \, \vec{\delta}_t \, \vec{v}_t) \,, \qquad (2) \\ \text{where} \quad \vec{v}_t &\equiv \frac{\partial \vec{w}_t}{\partial \ln \vec{p}} \end{aligned}$$

and μ is a global meta-learning rate. This approach rests on the assumption that each element of $\vec{p}$ affects $f_{\vec{w}}(\vec{x})$ only through the corresponding element of $\vec{w}$. With considerable variation, (2) forms the basis of most local rate adaptation methods found in the literature.

In order to avoid an expensive exponentiation [23] for each weight update, we typically use the linearization $e^u \approx 1 + u$, valid for small $|u|$, giving

$$\vec{p}_t = \vec{p}_{t-1} \max(\varrho, \, 1 + \mu \, \vec{\delta}_t \, \vec{v}_t) \,, \qquad (3)$$

where we constrain the multiplier to be at least (typically) $\varrho = 0.1$ as a safeguard against unreasonably small — or negative — values. For the meta-level gradient descent to be stable, μ must in any case be chosen such that the multiplier for $\vec{p}$ does not stray far from unity; under these conditions we find the linear approximation (3) quite sufficient.

Definition of $\vec{v}$. The *gradient trace* $\vec{v}$ should accurately measure the effect that a change in local learning rate has on the corresponding weight. It is tempting to consider only the *immediate* effect of a change in $\vec{p}_t$ on $\vec{w}_{t+1}$: declaring $\vec{w}_t$ and $\vec{\delta}_t$ in (1) to be independent of $\vec{p}_t$, one then quickly arrives at $\vec{v}_{t+1} = \vec{p}_t \, \vec{\delta}_t$ — *cf.* [9, 10, 11, 19].

However, this fails to take into account the incremental nature of gradient descent: a change in $\vec{p}$ affects not only the current update of $\vec{w}$, but also all future ones. Some authors account for this by setting $\vec{v}$ to an exponential average of past gradients [7, 9, 18]; in Section 4 we find empirically that the method of Almeida *et al.* [19] can indeed be improved by this approach. While such averaging serves to reduce the stochasticity of the product $\vec{\delta}_t \, \vec{\delta}_{t-1}$, the average remains one of immediate, single-step effects.

By contrast, Sutton [5] models the long-term effect of $\vec{p}$ on future weight updates in a linear system by carrying the relevant partials forward through time (*cf.* real-time recurrent learning [24]). This results in an iterative update rule for $\vec{v}$, which we extend here to nonlinear systems. As before, we differentiate (1) with respect to $\ln \vec{p}$, but we now consider the change in $\vec{p}$ to have occurred arbitrarily far in the past, giving

$$\begin{aligned} \vec{v}_{t+1} &= \frac{\partial \vec{w}_t}{\partial \ln \vec{p}} + \frac{\partial (\vec{p}_t \, \vec{\delta}_t)}{\partial \ln \vec{p}} \\ &= \vec{v}_t + \vec{p}_t \, \vec{\delta}_t - \vec{p}_t \left[\frac{\partial^2 f_{\vec{w}_t}(\vec{x}_t)}{\partial \vec{w} \, \partial \vec{w}^T} \, \frac{\partial \vec{w}_t}{\partial \ln \vec{p}} \right] \\ &= \vec{v}_t + \vec{p}_t \, (\vec{\delta}_t - H_t \vec{v}_t) \,, \qquad (4) \end{aligned}$$

where H_t denotes the instantaneous Hessian of $f_{\vec{w}}(\vec{x})$ at time t. Note that there is an efficient $O(n)$ algorithm to calculate $H_t \vec{v}_t$ without ever having to compute or store the matrix H_t itself [25].

Meta-level conditioning. The gradient descent in $\vec{p}$ at the meta-level (2) may of course suffer from ill-conditioning as much as the descent in $\vec{w}$ at the main level (1); the meta-descent in fact *squares* the condition number when $\vec{v}$ is defined as the previous gradient, or an exponential average of past gradients. Special measures to improve conditioning are thus required to make meta-descent work in non-trivial systems.

In many cases these take the form of squashing functions such as cosine [8] or sign

[9, 10, 11] used to radically normalize the $\vec{p}$-update. Unfortunately such nonlinearities do not preserve the zero-mean property that characterizes stochastic gradients in equilibrium — in particular, they will translate any skew in the equilibrium distribution into a non-zero mean change in $\vec{p}$. This causes convergence to non-optimal step sizes, and renders such methods unsuitable for online learning. Notably, Almeida *et al.* [19] avoid this pitfall by using a running estimate of the gradient's stochastic variance as their meta-normalizer (*cf.* [4]).

In addition to modeling the long-term effect of a change in local learning rate, our iterative gradient trace serves as a highly effective conditioner for the meta-descent (2): the fixpoint of (4) is given by

$$\vec{v}_t \;=\; H_t^{-1}\vec{\delta}_t \tag{5}$$

— a stochastic Newton step, which scales with the inverse of the gradient. Consequently, we can expect the product $\vec{\delta}_t\vec{v}_t$ in (2) to be a very well-conditioned quantity. Experiments confirm that SMD does not require explicit meta-level normalization.

3 Special Cases

The SMD framework generalizes many existing and novel step size adaptation methods. Setting $H_t \equiv \lambda I$ in (4) turns $\vec{v}$ back into an exponential average of past gradients [9, 18]. Additional omission of the λ and $\vec{p}_t$ factors reproduces the unnormalized method of Almeida *et al.* [19].

Applying (4) to the linear (LMS) system

$$\begin{aligned} f_{\vec{w}_t}(\vec{x}_t, y_t) &\equiv \frac{1}{2}\,(y_t - a_t)^2, \\ \text{where}\quad a_t &\equiv \vec{w}_t^{\,T}\vec{x}_t\,, \end{aligned} \tag{6}$$

diagonalizing the Hessian to $H_t = \mathrm{diag}(\vec{x}_t^{\,2})$, and adding a positive-bounding operation yields Sutton's IDBD algorithm [20]:

$$\vec{v}_{t+1} \;=\; \vec{v}_t\,(1 - \vec{p}_t\,\vec{x}_t^{\,2})^+ + \;\vec{p}_t\,\vec{\delta}_t \tag{7}$$

For a normalized LMS system that uses

$$\vec{w}_{t+1} \;=\; \vec{w}_t \;+\; k_t\,\vec{p}_t\,\vec{\delta}_t \tag{8}$$

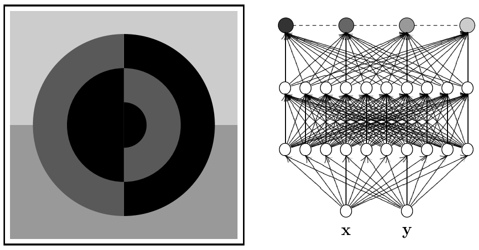

Figure 1: The four regions benchmark (left), and the neural network trained on it (right).

$$\text{where}\quad k_t \;\equiv\; \frac{1}{1 + \vec{x}_t^{\,T}(\vec{p}_t\,\vec{x}_t)} \tag{9}$$

we obtain Sutton's K1 algorithm [5]:

$$\vec{v}_{t+1} \;=\; (\vec{v}_t + k_t\,\vec{p}_t\,\vec{\delta}_t)(1 - k_t\,\vec{p}_t\,\vec{x}_t^{\,2}) \tag{10}$$

Our ELK1 algorithm [26] extends K1 by adding a nonlinear function σ to the system's output:

$$\begin{aligned} f_{\vec{w}_t}(\vec{x}_t, y_t) &\equiv \frac{1}{2}\,[\,y_t - \sigma(a_t)]^2 \\ k_t &\equiv \frac{1}{1 + \vec{x}_t^{\,T}(\vec{p}_t\,\vec{x}_t)\,\sigma'(a_t)^2} \\ \vec{v}_{t+1} &= (\vec{v}_t + k_t\,\vec{p}_t\,\vec{\delta}_t)[\,1 - k_t\,\vec{p}_t\,\vec{x}_t^{\,2}\sigma'(a_t)^2] \end{aligned} \tag{11}$$

The ELK1 update can be applied independently to each node in a multi-layer perceptron, providing a computationally attractive diagonal approximation to the full Hessian update for $\vec{v}$ given in (4).

4 Benchmark Results

We evaluated our work on the "four regions" classification task [27], a well-known benchmark problem [17, 28, 29, 30]: a fully connected feedforward network with 2 hidden layers of 10 units each (tanh nonlinearity) is to classify two continuous inputs (range [-1,1]) as illustrated in Figure 1. We used "softmax" output units and a negated cross-entropy loss function E.

We compared six online algorithms: SMD – equations (1), (3), and (4), ELK1 – equations (3), (8), and (11), ALAP – the normalized step size adaptation method of Almeida

Table 1: Computational cost of algorithms.

Algorithm	storage / param.	flops / update	cpu ms / pattern
mom.	2	8	7.3
vario-η	4	18	8.6
ALAP	3	15	8.0
S-ALAP	3	18	8.5
ELK1	3	19	9.9
SMD	3	22	10.3

et al. [19], S-ALAP – like ALAP, but smoothed by using an exponential trace of past gradients for $\vec{v}$, vario-η – a learning rate normalization method [4], and "momentum" – stochastic gradient descent with a fixed learning rate and momentum. Table 1 compares the computational cost associated with each algorithm in terms of how many floating-point numbers must be stored per weight parameter, how many floating-point operations must be performed per weight update, and how many milliseconds of CPU time were consumed per training pattern in our prototype implementation (written in Octave 2.0.10, running under Linux 2.0.29 on Intel Pentium Pro at 240 MHz).

The free parameters of each method were tuned for best reliable performance at the end of a run. This was easier for SMD and ELK1, which have only initial and the meta-learning rates as free parameters, than for the ALAP methods, which in addition require a smoothing rate parameter. Table 2 lists the values we settled on.

We trained each algorithm 25 times on a uniformly random sequence of 50 000 patterns, starting from different sets of

Table 2: Values used for the free parameters.

Parameter: / Algorithm:	(initial) lrng. rate	meta- lrng. rate	smoothing rate
mom., vario-η	0.01	—	0.95
(S-)ALAP	(0.10)	0.01	0.95
ELK1	(0.10)	0.10	—
SMD	(0.05)	0.05	—

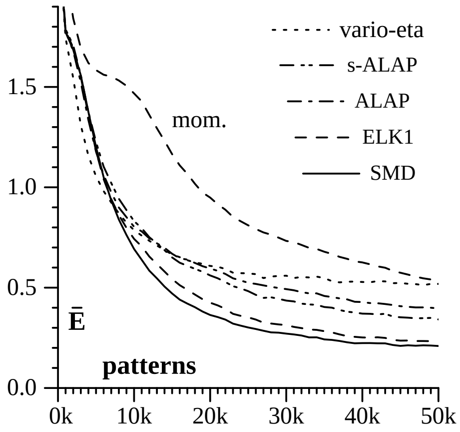

Figure 2: The average smoothed loss $\bar{E}$ during online training on the four regions task, plotted against the number of training patterns (uniformly random sampling).

uniformly random initial weights (range $[-0.3, 0.3]$; biases initialized to zero). Since each pattern was seen only once, the empirical loss E provided an unbiased estimate of expected loss, and hence generalization ability. Figure 2 shows the average value of $\bar{E}$ — an exponential trace (smoothing parameter 0.999) of E — during training.

While all accelerated methods conveyed a similar initial increase in convergence speed over gradient descent with momentum, there were clear differences in their later performance. The full Hessian update method of SMD converged most rapidly, followed closely by ELK1, the diagonalized, node-decoupled version. Both clearly outperformed ALAP, for which our smoothed

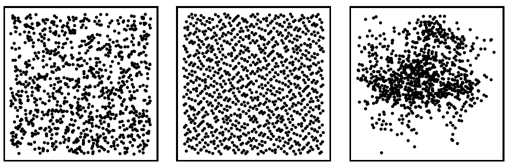

Figure 3: The first 1000 points of: a uniformly random sample (left), the Sobol sequence (center), and a Brownian random walk with standard deviation 0.1 (right).

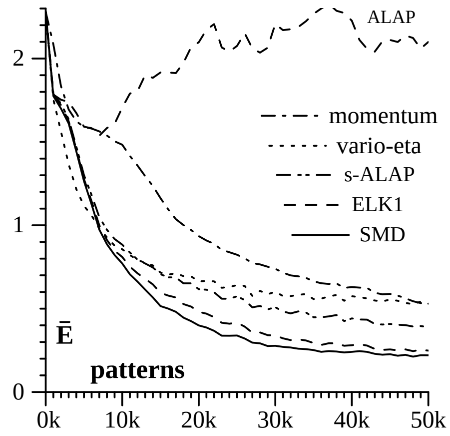

Figure 4: As Figure 2, but with patterns sampled according to the Sobol sequence.

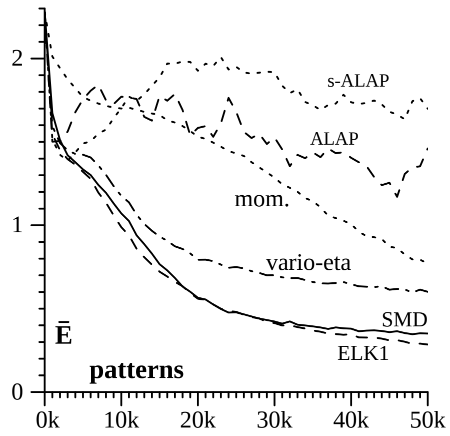

Figure 5: As Figure 2, but with training patterns sampled according to a Brownian random walk with standard deviation 0.1.

variant converged faster than Almeida *et al.*'s original formulation. Finally, vario-η exhibited the characteristic behavior of a normalization method, combining the best initial with the worst asymptotic performance. (We surmise that the asymptotic performance of vario-η could be improved by incorporating a global learning rate adaptation mechanism.)

Sobol sampling. In contrast to our methods, ALAP is derived under the assumption that successive training patterns are statistically independent — a condition that may be hard to meet *e.g.* in *situated* applications, where the input is provided by an environment over which the learner does not have complete control. We repeated the above experiment while sampling the input space according to the Sobol sequence [31, pp. 311–314], which achieves a super-uniform covering (Figure 3, center) at the expense of introducing strong short-term correlations between samples. Figure 4 shows that while the other methods remained largely unaffected, ALAP failed to converge in all 25 runs. We tried varying its free parameters, but found that convergence was restored only by essentially turning off gain adaptation ($\mu \to 0$). Our smoothed variant S-ALAP, however, was able to cope with this sampling strategy.

Brownian sampling. It may be argued that the Sobol sequence does not feature the kind of dependencies between input samples that are encountered in practice. We therefore repeated our experiments once more with the input space sampled by a Brownian random walk with standard deviation 0.1 (Figure 3, right), representative of the way a situated agent might go about exploring its environment. This introduces not only strong correlations, but also a notion of locality — and hence non-stationarity — into the sampling process.

Now exponential averaging of the gradient trace no longer suffices to smooth out the correlations: while ALAP's weights diverged in all but a single run, our smoothed variant, S-ALAP, also diverged about half the time. This casts some doubt on ALAP's reliability in situations where statistical independence between successive input samples cannot be guaranteed.

The other methods converged reliably (Figure 5), except that SMD diverged twice. We consider this quite acceptable, given the speed with which the algorithm converged in the remaining 23 runs. Interestingly though, ELK1 appears to slightly outperform SMD in this case, for as yet unknown reasons.

Stability. We noticed empirically that the stochastic gradient trace (4) used by SMD may occasionally diverge, in particular if the meta-learning μ has been chosen too high or too low. While it is easy to prevent such divergence in an ad-hoc fashion — for instance, by zeroing all elements of $H_t\vec{v}_t$ whose sign differs from that of the corresponding element of $\vec{v}_t$ — this typically also compromises the rapid convergence of the algorithm. We are now investigating this phenomenon, with the goal of finding a way to ensure the stability of SMD while preserving its remarkable speed of convergence.

Acknowledgment

This work was supported by the Swiss National Science Foundation under grant number 2000–052678.97/1.

References

[1] N. N. Schraudolph, "Local gain adaptation in stochastic gradient descent", in *Proc. Intl. Conf. Artificial Neural Networks*, Edinburgh, Scotland, 1999, pp. 569–574, IEE, London, `http://n.schraudolph.org/pubs/smd.ps.gz`.

[2] S. Becker and Y. LeCun, "Improving the convergence of back-propagation learning with second order methods", in *Proceedings of the 1988 Connectionist Models Summer School*, D. Touretzky, G. Hinton, and T. Sejnowski, Eds., Pittsburg 1988, 1989, pp. 29–37, Morgan Kaufmann, San Mateo.

[3] N. N. Schraudolph and T. J. Sejnowski, "Tempering backpropagation networks: Not all weights are created equal", in *Advances in Neural Information Processing Systems*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds. 1996, vol. 8, pp. 563–569, The MIT Press, Cambridge, MA, `http://n.schraudolph.org/pubs/nips95.ps.gz`.

[4] R. Neuneier and H. G. Zimmermann, "How to train neural networks", in *Neural Networks: Tricks of the Trade*, vol. 1524 of *Lecture Notes in Computer Science*, pp. 373–423. Springer Verlag, Berlin, 1998.

[5] R. S. Sutton, "Gain adaptation beats least squares?", in *Proc. 7th Yale Workshop on Adaptive and Learning Systems*, 1992, pp. 161–166, `ftp://ftp.cs.umass.edu/pub/anw/pub/sutton/sutton-92b.ps.gz`.

[6] Y. LeCun, P. Y. Simard, and B. Pearlmutter, "Automatic learning rate maximization in large adaptive machines", in *Advances in Neural Information Processing Systems*, S. J. Hanson, J. D. Cowan, and C. L. Giles, Eds. 1993, vol. 5, pp. 156–163, Morgan Kaufmann, San Mateo, CA.

[7] N. Murata, K.-R. Müller, A. Ziehe, and S.-i. Amari, "Adaptive on-line learning in changing environments", in *Advances in Neural Information Processing Systems*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. 1997, vol. 9, pp. 599–605, The MIT Press, Cambridge, MA.

[8] L.-W. Chan and F. Fallside, "An adaptive training algorithm for back propagation networks", *Computer Speech and Language*, **2**:205–218, 1987.

[9] R. Jacobs, "Increased rates of convergence through learning rate adaptation", *Neural Networks*, **1**:295–307, 1988.

[10] T. Tollenaere, "SuperSAB: fast adaptive back propagation with good scaling properties", *Neural Networks*, **3**:561–573, 1990.

[11] F. M. Silva and L. B. Almeida, "Speeding up back-propagation", in *Advanced Neural Computers*, R. Eckmiller, Ed., Amsterdam, 1990, pp. 151–158, Elsevier.

[12] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm", in *Proc. International Conference on Neural Networks*, San Francisco, CA, 1993, pp. 586–591, IEEE, New York.

[13] M. Riedmiller, "Advanced supervised learning in multi-layer perceptrons — from backpropagation to adaptive learning algorithms", *Computer Standards & Interfaces*, **16**:265–278, 1994.

[14] A. Lapedes and R. Farber, "A self-optimizing, nonsymmetrical neural net for content addressable memory and pattern recognition", *Physica*, **D 22**:247–259, 1986.

[15] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, "Ac-

celerating the convergence of the back-propagation method", *Biological Cybernetics*, **59**:257–263, 1988.

[16] R. Battiti, "Accelerated back-propagation learning: Two optimization methods", *Complex Systems*, **3**:331–342, 1989.

[17] X.-H. Yu, G.-A. Chen, and S.-X. Cheng, "Acceleration of backpropagation learning using optimised learning rate and momentum", *Electronics Letters*, **29**(14):1288–1290, 8 July 1993.

[18] M. E. Harmon and L. C. Baird, III, "Multi-player residual advantage learning with general function approximation", Tech. Rep. WL-TR-1065, Wright Laboratory, WL/AACF, Wright-Patterson Air Force Base, OH 45433-7308, 1996, `http://www.leemon.com/papers/sim_tech/sim_tech.pdf`.

[19] L. B. Almeida, T. Langlois, J. D. Amaral, and A. Plakhov, "Parameter adaptation in stochastic optimization", in *On-Line Learning in Neural Networks*, D. Saad, Ed., Publications of the Newton Institute, chapter 6, pp. 111–134. Cambridge University Press, 1999, `ftp://146.193.2.131/pub/lba/papers/adsteps.ps.gz`.

[20] R. S. Sutton, "Adapting bias by gradient descent: an incremental version of delta-bar-delta", in *Proc. 10th National Conference on Artificial Intelligence*. 1992, pp. 171–176, The MIT Press, Cambridge, MA, `ftp://ftp.cs.umass.edu/pub/anw/pub/sutton/sutton-92a.ps.gz`.

[21] J. Kivinen and M. K. Warmuth, "Exponentiated gradient versus gradient descent for linear predictors", Tech. Rep. UCSC-CRL-94-16, University of California, Santa Cruz, June 1994.

[22] J. Kivinen and M. K. Warmuth, "Additive versus exponentiated gradient updates for linear prediction", in *Proc. 27th Annual ACM Symposium on Theory of Computing*, New York, NY, May 1995, pp. 209–218, The Association for Computing Machinery.

[23] N. N. Schraudolph, "A fast, compact approximation of the exponential function", *Neural Computation*, **11**(4):853–862, 1999, `http://n.schraudolph.org/pubs/exp.ps.gz`.

[24] R. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks", *Neural Computation*, **1**:270–280, 1989.

[25] B. A. Pearlmutter, "Fast exact multiplication by the Hessian", *Neural Computation*, **6**(1):147–160, 1994.

[26] N. N. Schraudolph, "Online local gain adaptation for multi-layer perceptrons", Tech. Rep. IDSIA-09-98, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Galleria 2, CH-6928 Manno, Switzerland, 1998, `http://n.schraudolph.org/pubs/olga.ps.gz`.

[27] S. Singhal and L. Wu, "Training multilayer perceptrons with the extended Kalman filter", in *Advances in Neural Information Processing Systems. Proceedings of the 1988 Conference*, D. S. Touretzky, Ed., San Mateo, CA, 1989, pp. 133–140, Morgan Kaufmann.

[28] G. V. Puskorius and L. A. Feldkamp, "Decoupled extended Kalman filter training of feedforward layered networks", in *Proceedings of the International Joint Conference on Neural Networks*, Seattle, WA, 1991, vol. I, pp. 771–777, IEEE.

[29] S. Shah, F. Palmieri, and M. Datum, "Optimal filtering algorithms for fast learning in feedforward neural networks", *Neural Networks*, **5**:779–787, 1992.

[30] E. S. Plumer, "Training neural networks using sequential-update forms of the extended Kalman filter", Tech. Rep. LA-UR-95-422, Los Alamos National Laboratory, 1995.

[31] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, second edition, 1992.