Online Learning with Adaptive Local Step Sizes

Nicol N. Schraudolph
nic@idsia.ch

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

Abstract

Almeida *et al.* have recently proposed *online* algorithms for local step size adaptation in nonlinear systems trained by gradient descent. Here we develop an alternative to their approach by extending Sutton's work on linear systems to the general, nonlinear case. The resulting 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 as well as stochastic gradient descent with fixed learning rate and momentum.

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 rates for rapid convergence. *Normalization* methods [1, 2, 3] *calculate* the optimal local step size 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 improve them [4].

Unfortunately most of the existing step size adaptation algorithms for neural networks adapt only a single, global learning rate [5, 6], can be used only in batch training [7, 8, 9, 10, 11], or both. Given the well-known advantages of stochastic gradient descent, it would be desirable to have online methods for adapting local step sizes. The first such techniques that we are aware of have recently been proposed by Almeida *et al.* [12]. Here we develop an alternative to their approach by extending work by Sutton [4, 13] from linear systems to the general, nonlinear case.

2 Basic 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:

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

The local learning rates $\vec{p}$ are best adapted by exponentiated gradient descent [14], so that they can cover a wide dynamic range while staying strictly positive:

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

and μ is a global meta-learning rate. The definition of $\vec{v}$ 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 [15] for each weight update, we typically use the linearization $e^u \approx 1 + u$, valid for small $|u|$, giving

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

where we constrain the multiplier to be at least (typically) $\varrho = 0.1$ as a safeguard against unreasonably small — or negative — values.

Definition of $\vec{v}$. It is tempting to look at (1), declare $\vec{w}_t$ and $\vec{\delta}_t$ to be independent of $\vec{p}_t$, and quickly arrive at $\vec{v}_{t+1} = \vec{p}_t \, \vec{\delta}_t$, which (up to normalization) is the approach of Almeida *et al.* [12]. 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 future ones. Some authors account for this by setting $\vec{v}$ to an exponential average of past gradients [6, 8]. We prefer to extend Sutton's exact method [4] for iterative calculation of $\vec{v}$ to nonlinear systems:

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

where H_t denotes the instantaneous Hessian of $f_{\vec{w}}(\vec{x})$ at time t. Note that there is an efficient $O(n)$ method to calculate the matrix-vector product $H_t \vec{v}_t$ without ever computing or storing H_t itself [16].

Meta-level normalization. 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); in a sense we have only pushed the problem of choosing the appropriate step size down one meta-level. In many step size adaptation algorithms, this issue is addressed by incorporating squashing functions such as cosine [7] or sign [8, 9, 10] in the update for $\vec{p}$. Unfortunately such nonlinearities do not preserve the zero-mean property of a skewed density; this makes these methods unsuitable for online learning, whose equilibrium is characterized by zero-mean, skewed noise in the stochastic gradient. Notably, Almeida *et al.* [12] avoid this pitfall by using a running estimate of the gradient's noise variance as their normalizer (*cf.* [3]).

Our iterative definition of $\vec{v}$ has the advantage of serving 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 our algorithm does not require (and in fact is hampered by) additional meta-level normalization as in [12].

3 Special Cases

The $\vec{v}$ update introduced above forms the basis for a number of existing and novel step size adaptation methods. Simplifying (4) by setting $H_t = \lambda I$ produces algorithms closely related to those of Almeida *et al.* [12, $\lambda = 1$] and Jacobs [8], respectively. Applying (4) to the linear (LMS) system

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

diagonalizing the instantaneous Hessian $H_t = \vec{x}_t\vec{x}_t^{\,T}$ to $H_t \approx \operatorname{diag}(\vec{x}_t^{\,2})$, and adding a positive-bounding operation, yields Sutton's IDBD algorithm [13]:

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

For a normalized LMS system, the same approach produces his K1 algorithm [4]. Our ELK1 algorithm [17] extends K1 by adding a sigmoid function σ to the system's output, giving

$$\begin{aligned} f_{\vec{w}_t}(\vec{x}_t, y_t) \;&\equiv\; \frac{1}{2}\,(y_t - \sigma(a_t))^2, \qquad \vec{w}_{t+1} \;=\; \vec{w}_t \;+\; k_t\,\vec{p}_t\,\vec{\delta}_t\,, \\ \vec{v}_{t+1} \;&=\; (\vec{v}_t \;+\; k_t\,\vec{p}_t\,\vec{\delta}_t)\,[\,1 \;-\; k_t\,\vec{p}_t\,\vec{x}_t^{\,2}\,\sigma'(a_t)^2\,]\,, \\ &\qquad \text{where} \quad k_t \;\equiv\; \frac{1}{1 + \vec{x}_t^{\,T}(\vec{p}_t\,\vec{x}_t)\,\sigma'(a_t)^2} \end{aligned} \tag{8}$$

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).

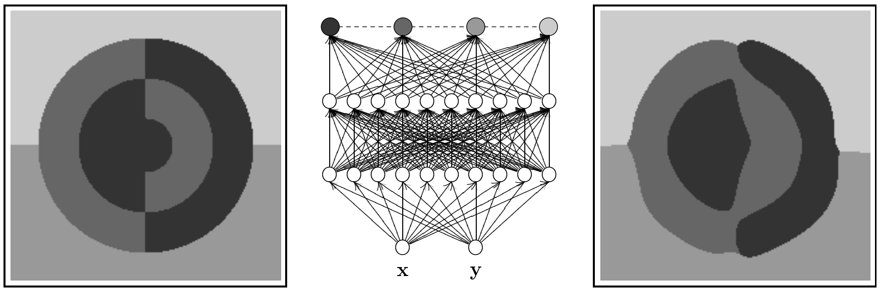

Figure 1: The four regions classification task (left), the neural network trained on it (center), and a typical solution after 50 000 training patterns (right).

4 Benchmark Results

We evaluated our approach on the "four regions" classification task [18]. 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 five online algorithms: "SMD" — equations (1), (3), and (4), "ELK1" — equations (3) and (8), "ALAP" — the normalized step size adaptation method of Almeida *et al.* [12], "vario-η" — a learning rate normalization method [3], and "momentum" — stochastic gradient descent with a fixed learning rate and momentum. The free parameters of each method were tuned for best performance; this was easier for SMD and ELK1, which have only the initial and meta-learning rates as free parameters, than for ALAP, which in addition requires a smoothing rate constant.

We trained each algorithm 25 times on a uniformly random sequence of 50 000 patterns, starting from different sets of 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 (left) shows the average value of $\bar{E}$ — an exponential trace (smoothing parameter 0.999) of E — during training. While all accelerated methods convey a similar initial increase in convergence speed over gradient descent with momentum, there are clear differences in their later performance. Our full Hessian update method (SMD) converges slightly but reliably faster than ELK1, the diagonalized, node-decoupled version. Both clearly outperform ALAP, the only other local step size adaptation method for online learning that we know of. Finally, vario-η behaves like a typical normalization method, combining the best initial with the worst asymptotic performance. (We surmise that this shortcoming of vario-η could be addressed by combining it with a global learning rate adaptation mechanism.)

There are other differences between the above algorithms. For instance, ALAP is derived under the assumption that successive training patterns are

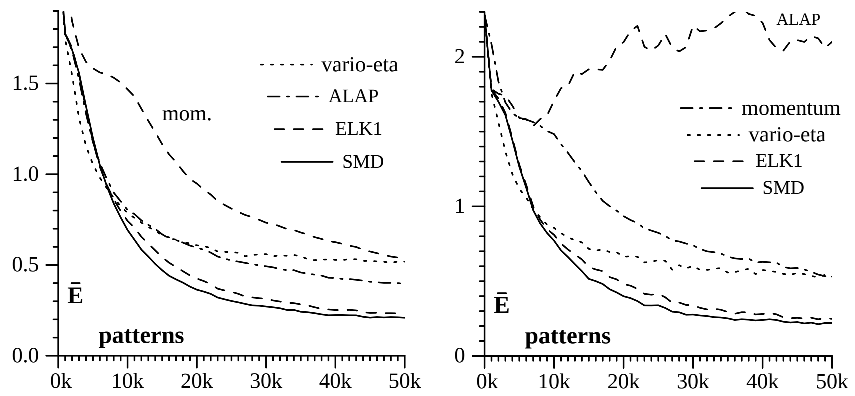

Figure 2: The average smoothed loss $\bar{E}$ during online training on the four regions task, plotted against the number of training patterns, for uniformly random (left) *vs.* super-uniform Sobol sampling (right).

statistically independent — a condition that may be hard to meet *e.g.* in *situated* applications, where the input is provided by a real-time 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 [19, pp. 311–314], which achieves a super-uniform covering of the input space at the expense of introducing strong short-term correlations between samples. Figure 2 (right) shows that while the other methods remained largely unaffected, ALAP failed to converge in all 25 runs. This casts some doubt on ALAP's reliability in situations where independence between successive training patterns cannot be guaranteed.

References

[1] 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.

[2] 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, `ftp://ftp.idsia.ch/pub/nic/nips95.ps.gz`.

[3] 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.

[4] 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`.

[5] 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.

[6] 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.

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

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

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

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

[11] 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.

[12] 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. Cambridge University Press, 1999, `ftp://146.193.2.131/pub/lba/papers/adsteps.ps.gz`.

[13] 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`.

[14] 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.

[15] N. N. Schraudolph, "A fast, compact approximation of the exponential function", *Neural Computation*, **11**(4):853–862, 1999, `ftp://ftp.idsia.ch/pub/nic/exp.ps.gz`.

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

[17] N. N. Schraudolph, "Online local gain adaptation for multi-layer perceptrons", Tech. Rep. IDSIA-09-98, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Corso Elvezia 36, 6900 Lugano, Switzerland, 1998, `ftp://ftp.idsia.ch/pub/nic/olga.ps.gz`.

[18] 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.

[19] 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.