Stable Adaptive Momentum for Rapid Online Learning in Nonlinear Systems*

Thore Graepel and Nicol N. Schraudolph

Institute of Computational Science
ETH Zürich, Switzerland
{graepel,schraudo}@inf.ethz.ch

Abstract We consider the problem of developing rapid, stable, and scalable stochastic gradient descent algorithms for optimisation of very large nonlinear systems. Based on earlier work by Orr et al. on adaptive momentum—an efficient yet extremely unstable stochastic gradient descent algorithm—we develop a stabilised adaptive momentum algorithm that is suitable for noisy nonlinear optimisation problems. The stability is improved by introducing a forgetting factor $0 \leq \lambda \leq 1$ that smoothes the trajectory and enables adaptation in non-stationary environments. The scalability of the new algorithm follows from the fact that at each iteration the multiplication by the curvature matrix can be achieved in $O(n)$ steps using automatic differentiation tools. We illustrate the behaviour of the new algorithm on two examples: a linear neuron with squared loss and highly correlated inputs, and a multilayer perceptron applied to the four regions benchmark task.

1 Introduction

Optimisation problems arise in a wide variety of fields including science, engineering, and business. We focus on optimisation algorithms for problems with a twice-differentiable objective function that are large in scale, non-linear, non-convex, that might be non-stationary, and for which only noisy measurements of the objective function value and its first and second derivatives w.r.t. the parameters are available. This is the realm of online stochastic gradient methods.

2 Adaptive Momentum

We consider the problem of minimising a function $f : \mathbb{R}^n \to \mathbb{R}$ with respect to its parameters $\mathbf{w} \in \mathbb{R}^n$. In online optimisation the weight vector $\mathbf{w}_t$ at time step t is updated according to

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{v}_t \,. \tag{1}$$

The simplest form of gradient descent is recovered by choosing $\mathbf{v}_t := -\mu \mathbf{g}_t$ where μ is a scalar learning rate parameter and $\mathbf{g}_t := \nabla_{\mathbf{w}} f_t(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_t}$ is the

* Proc. Intl. Conf. Artificial Neural Networks, LNCS, Springer Verlag, Berlin 2002

instantaneous gradient measurement of the objective function f with respect to the parameters $\mathbf{w}$ evaluated at $\mathbf{w}_t$.

Various improvements over this simple scheme are conceivable. Decreasing $\mu := \mu_t$ deterministically as a function of time aims at quickly traversing the search space at the beginning of the optimisation and to smoothly converge at the end. Replacing μ by a diagonal matrix allows the use of individual learning rates for each parameter. Replacing μ by a full matrix makes it possible to include local curvature information into the optimisation process.

We follow a different line of development that suggests to introduce a momentum term,

$$\mathbf{v}_t := -\mu\left(\rho\mathbf{g}_t - \mathbf{v}_{t-1}\right) . \tag{2}$$

The intuition behind this update rule is to stabilise the trajectory through search space by choosing the new update $\mathbf{v}_t$ as a linear combination of the previous update $\mathbf{v}_{t-1}$ and the new gradient $\mathbf{g}_t$. The adaptive version of this idea, incorporating a curvature matrix $\mathbf{C}$ into the update, has been introduced in [2,4],

$$\mathbf{v}_t := -\mu_t\left(\rho_t\mathbf{g}_t - \left(\mathbf{C}_t + \frac{1}{\mu_t}\mathbf{I}\right)\mathbf{v}_{t-1}\right) , \tag{3}$$

where $\mu_t < \lambda_{\max}^{-1}\left(\mathbf{C}_t\right)$ and ρ_t is annealed from 1 down to 0. Let us assume $\mu_t := \mu$, $\rho_t = \rho$, $\mathbf{g}_t := \mathbf{g}$, and $\mathbf{C}_t := \mathbf{C}$ constant for the sake of argument. Setting $\mathbf{v}_t = \mathbf{v}_{t-1} =: \mathbf{v}_\infty$ and solving for $\mathbf{v}_\infty$ it is easily seen that the update rule (3) has a fixed-point at

$$\mathbf{v}_\infty = -\rho\mathbf{C}^{-1}\mathbf{g} . \tag{4}$$

Thus, using $\mathbf{v}_\infty$ instead of $\mathbf{v}_t$ in (1) and setting $\rho := 1$ we recover Newton's methods if $\mathbf{C}$ is the Hessian, $\mathbf{C}_t := \mathbf{H}_t := \nabla^2_{\mathbf{w}} f_t\left(\mathbf{w}\right)\big|_{\mathbf{w}=\mathbf{w}_t}$. For nonlinear problems it is advisable to use the Gauss-Newton approximation to the Hessian which is guaranteed to be positive semidefinite while retaining as much information as possible [6].

3 Stable Adaptive Momentum

The above algorithm turns out to be very efficient, but unstable for practical non-convex problems such as multilayer perceptrons. As a stabilising mechanism we suggest to use an exponential average over past values of $\mathbf{v}_t$ governed by a forgetting factor $0 \leq \lambda \leq 1$, resulting in an update rule

$$\mathbf{v}_t := -\mu_t\left(\rho_t\mathbf{g}_t - \lambda\left(\mathbf{C}_t + \frac{1}{\mu_t}\mathbf{I}\right)\mathbf{v}_{t-1}\right) . \tag{5}$$

Clearly, for $\lambda = 1$ we recover the matrix momentum update (3) and for $\lambda = 0$ standard gradient descent. Again assuming $\mu_t := \mu$, $\rho_t = \rho$, $\mathbf{g}_t := \mathbf{g}$, and $\mathbf{C}_t := \mathbf{C}$ constant, the new update rule (5) has a fixed point at

$$\mathbf{v}_\infty = -\rho\left(\lambda\mathbf{C} + (1-\lambda)\frac{1}{\mu}\mathbf{I}\right)^{-1}\mathbf{g} . \tag{6}$$

Thus in the limit the effective curvature matrix is a convex combination of $\mathbf{C}$ and $\lambda_{\max}(\mathbf{C})\,\mathbf{I}$, comparable to the Levenberg-Marquardt approach [3]. In particular, if we choose $\frac{1}{\mu} := \lambda_{\max}(\mathbf{C})$ as suggested in [1] the condition number$\mathcal{N}(\mathbf{C}) := \lambda_{\max}(\mathbf{C}) / \lambda_{\min}(\mathbf{C})$ is improved to

$$\mathcal{N}\left(\lambda\mathbf{C} + (1-\lambda)\frac{1}{\mu}\mathbf{I}\right) = \frac{\lambda\,\lambda_{\max}(\mathbf{C}) + \frac{1-\lambda}{\mu}}{\lambda\,\lambda_{\min}(\mathbf{C}) + \frac{1-\lambda}{\mu}} = \frac{\lambda_{\max}(\mathbf{C})}{\lambda\,\lambda_{\min}(\mathbf{C}) + (1-\lambda)\,\lambda_{\max}(\mathbf{C})} \,.$$

The denominator of the condition number is proportional to a convex combination of the largest and smallest eigenvalues of $\mathbf{C}$.

For convenient and efficient implementation of this algorithm *automatic differentiation* tools[1] can be used to calculate gradient $\mathbf{g}_t$ and curvature matrix-vector product $\mathbf{C}_t\mathbf{v}$ in $\mathcal{O}(n)$ as described in [5]. The Gauss-Newton approximation of the Hessian should be used to ensure positive semidefiniteness [6].

Algorithm 1 Stable Adaptive Momentum (SAM)

Require: A twice-differentiable objective function $f : \mathbb{R}^n \to \mathbb{R}$
Require: instantaneous gradient $\mathbf{g}_t = \nabla_{\mathbf{w}} f_t(\mathbf{w})|_{\mathbf{w}=\mathbf{w}_t}$ and curvature matrix vector product $\mathbf{C}_t\mathbf{v}$
Require: Forgetting factor $0 \le \lambda \le 1$, learning rates $0 < \rho \le 1$ and $\mu > 0$, and objective function convergence criterion $\varepsilon_{\mathrm{f}} > 0$.
Ensure: Storage for $\mathbf{v}_{\text{new}}, \mathbf{v}_{\text{old}}, \mathbf{w}_{\text{new}}, \mathbf{w}_{\text{old}}$.
  Initialise $\mathbf{w}_{\text{old}}$ with small random numbers, and $\mathbf{v}_{\text{old}} \leftarrow g_0$
  repeat
    $\mathbf{v}_{\text{new}} \leftarrow -\mu\left(\rho\mathbf{g}_t - \lambda\left(\mathbf{C}_t + \frac{1}{\mu}\mathbf{I}\right)\mathbf{v}_{\text{old}}\right)$
    $\mathbf{w}_{\text{new}} \leftarrow \mathbf{w}_{\text{old}} + \mathbf{v}_{\text{new}}$
  until Time-averaged error $\langle |f_t(\mathbf{w}_t) - f_t(\mathbf{w}_t)| \rangle_t < \varepsilon_{\mathrm{f}}$

4 Numerical Experiments

4.1 The Quadratic Bowl

In order to study the influence of the parameter λ on the behaviour of the algorithm and to understand its interaction with the learning rate parameter ρ we considered the unconstrained quadratic optimisation problem of minimising the function

$$f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{H}\mathbf{w}\,, \quad \mathbf{H} = \mathbf{J}\,\mathbf{J}^{\mathcal{T}}\,, \quad \mathbf{J} = \begin{pmatrix} 2 & 1.6 \\ 1.2 & 3.2 \end{pmatrix}.$$

With eigenvalues of $\mathbf{H}$ given by 16.89 and 1.19 this configuration models the situation of long diagonal valleys in the objective function that often occur in neural network learning. Obviously the optimal solution is $\mathbf{w} = \mathbf{0}$ with $f(\mathbf{0}) = 0$. Samples for online learning are drawn from a normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{H})$.

[1] see `http://www-unix.mcs.anl.gov/autodiff/`

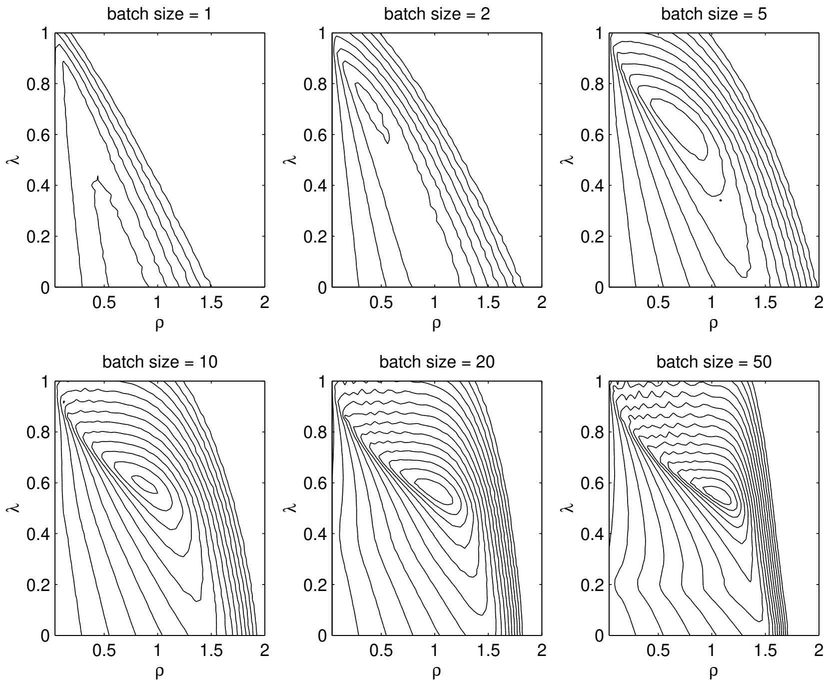

Figure 1. Log-contour plots of objective function value $f(\mathbf{w}) = \mathbf{w}^T \mathbf{H} \mathbf{w}$ after 100 steps of SAM, averaged over starting points $\{0.0, 0.01, \ldots, 4\pi\}$ on the unit circle, as a function of the ρ and λ parameters. Each contour corresponds to a 100-fold improvement in f.

Figure 1 shows log-contour plots of the value of the objective function $f(\mathbf{w}_t)$ after 100 update steps, averaged over different starting points on the unit circle at angles $\{0.0, 0.01, \ldots, 4\pi\}$, as a function of the ρ and λ parameters, for different batch sizes. When learning fully online (batch size 1), simple gradient descent ($\lambda = 0$) works best: in this regime the curvature measurements are so noisy that they do not accelerate convergence. For batch sizes ¿ 1, however, SAM clearly outperforms both simple gradient descent and standard adaptive momentum ($\lambda = 1$). (Note that each contour on the plot corresponds to a 100-fold improvment in objective function value.) As the batch size increases, the optimal regime emerges at $\lambda \approx 0.5 \ldots 0.6$, $\rho = 1$ — the latter value illustrating that correctly stabilized and conditioned (through λ and μ parameters, respectively) SAM in fact does not require the ρ parameter. Finally, the plot for batch size 50 shows how lowering λ serves to dampen oscillations observable (as a function of ρ) for high values of λ.

4.2 The Four Regions Task

As a more realistic and difficult test example we use the four regions classification task [7] to be solved by a multi-layer perceptron, as shown in Figure 2

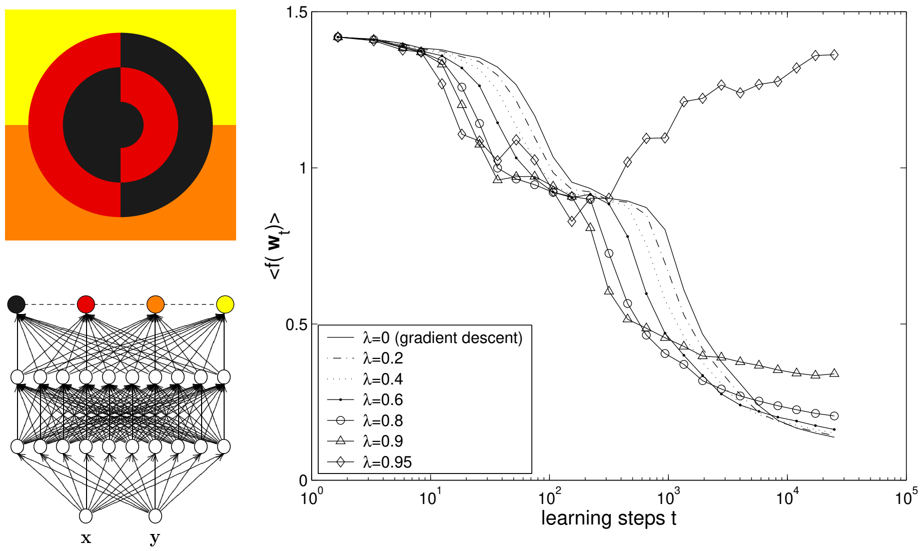

Figure 2. Four regions task (top left) and corresponding learning curves (right) using stable adaptive momentum with different values of λ applied to a 2-10-10-4 fully connected feed-forward neural network (bottom left) with tanh activation functions, softmax output, and cross-entropy loss function.

(left). The network has two input units, four output units and two layers of 10 hidden units (with tanh transfer function) each, with neighbouring layers fully connected, resulting in $n = 184$ weights. The classification is implemented through a softmax function applied to the outputs and a cross-entropy loss function. For each run the weights are initialised to uniformly random values in the interval $[-0.3, 0.3]$. Training patterns are generated online by drawing independent, uniformly random input samples; since each pattern is seen only once, the empirical loss provides an unbiased estimate of generalisation ability. Patterns are presented in mini-batches of ten each.

We compare SAM (with $\rho = 1$) to standard gradient descent with constant learning rate μ. The learning rate $\mu = 0.01$ was chosen such that standard gradient descent achieves the lowest value of $f(\mathbf{w})$ after 25 000 update steps. Figure 2 (right) shows learning curves for different values of λ in the range $[0.0, 0.95]$. To put these results in perspective, note that the original adaptive momentum algorithm (corresponding to $\lambda = 1$) diverges immediately everytime, as observed already in [4]. This can also be seen from the curve for $\lambda = 0.95$ in Figure 2 (right) that illustrates this problem of divergence for values greater than $\lambda = 0.9$. Compared to simple gradient descent ($\lambda = 0$), increasing λ generally results in a faster transient reduction of f while the asymptotic quality of the solution becomes worse.

We found that the ad-hoc annealing schedule $\lambda(t) = 0.9/(1 + \exp\frac{t-5000}{1000})$, which decreases λ logistically from 0.9 in the search phase to zero in the conver-

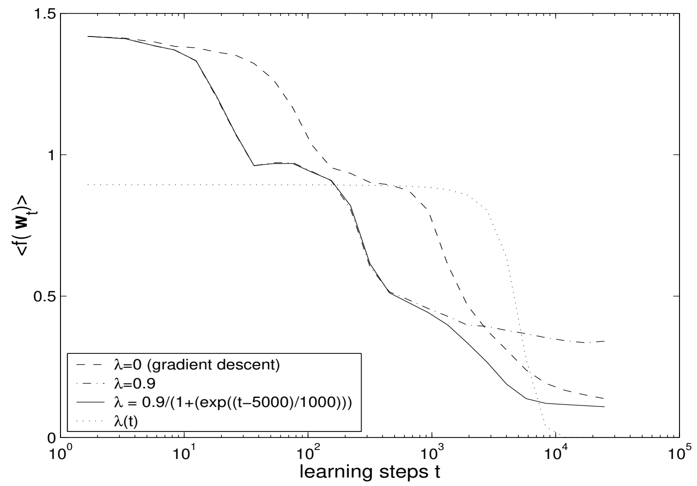

Figure 3. Learning curves for the four regions task illustrating an ad-hoc annealing scheme for λ that accelerates convergence but retains good asymptotic performance.

gence phase, is capable of combining the advantages of fast transient and good asymptotic performance (Figure 3). We are now investigating how to control λ adaptatively so as to automate this annealing process.

References

1. Y. LeCun, P. Y. Simard, and B. Pearlmutter. Automatic learning rate maximization in large adaptive machines. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 156–163. Morgan Kaufmann, San Mateo, CA, 1993.
2. T. K. Leen and G. B. Orr. Optimal stochastic search and adaptive momentum. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 477–484. Morgan Kaufmann, San Francisco, CA, 1994.
3. D. Marquardt. An algorithm for least-squares estimation of non-linear parameters. *Journal of the Society of Industrial and Applied Mathematics*, 11(2):431–441, 1963.
4. G. B. Orr. *Dynamics and Algorithms for Stochastic Learning.* PhD thesis, Department of Computer Science and Engineering, Oregon Graduate Institute, Beaverton, OR 97006, 1995. `ftp://neural.cse.ogi.edu/pub/neural/papers/orrPhDch1-5.ps.Z`, `orrPhDch6-9.ps.Z`.
5. B. A. Pearlmutter. Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147–160, 1994.
6. N. N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7), 2002. `http://www.inf.ethz.ch/~schraudo/pubs/mvp.ps.gz`.
7. S. Singhal and L. Wu. Training multilayer perceptrons with the extended Kalman filter. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems. Proceedings of the 1988 Conference*, pages 133–140, San Mateo, CA, 1989. Morgan Kaufmann.