The Dynamics of Matrix Momentum

Magnus Rattray and David Saad
Neural Computing Research Group, Aston University, Birmingham B4 7ET, UK.

Abstract

We analyse the matrix momentum algorithm, which provides an efficient approximation to on-line Newton's method, by extending a recent statistical mechanics framework to include second order algorithms. We study the efficacy of this method when the Hessian is available and also consider a practical implementation which uses a single example estimate of the Hessian. The method is shown to provide excellent asymptotic performance, although the single example implementation is sensitive to the choice of training parameters. We conjecture that matrix momentum could provide efficient matrix inversion for other second order algorithms.

1 Introduction

On-line learning is a popular method for training multilayer feed-forward neural networks in which the network parameters are updated according to only the latest in a sequence of training examples. Second order methods, which incorporate information about the curvature of the mean error surface, have been shown to be asymptotically optimal (e.g., [1, 2]) but they are often expensive both computationally and in terms of storage space; for example, they may require averaging over the entire data set to determine the Hessian followed by a matrix inversion. Orr & Leen [3, 4] have recently proposed a novel on-line algorithm which uses a momentum term with an adaptive matrix momentum parameter to approximate on-line Newton's method. They claim that this algorithm is asymptotically optimal and insensitive to the choice of external parameters.

The aim of this paper is twofold. We first employ a theoretical framework, recently developed for studying the dynamics of on-line learning [5], to study the performance of an idealized version of matrix momentum in which the exact Hessian is available. In practice, the Hessian is not available on-line and we therefore use the same theoretical framework to examine the performance using a single example approximation to the Hessian, as suggested by Orr & Leen [3], and consider its limitations. There are several advantages in conducting a theoretical study in the manner described here over a numerical one. Studying the average behaviour, using modest computational means, we perform an unbiased assessment of the algorithm which is insensitive to the choice of training examples. Moreover, having an analytical description of the system enables us to determine the optimal achievable asymptotic performance, which can then be used to assess the above algorithms. Combining these results with recent work on the asymptotic dynamics of gradient descent could provide analytical results for optimal and maximal learning parameters [6].

2 General framework

We consider a map from an N-dimensional input space $\boldsymbol{\xi} \in \Re^N$ onto a scalar, realized through a model $\sigma(\mathbf{J},\boldsymbol{\xi}) = \sum_{i=1}^{K} g(\mathbf{J}_i \cdot \boldsymbol{\xi})$, which can be viewed as a soft committee machine, where $g(x) \equiv \mathrm{erf}(x/\sqrt{2})$, is the activation function of the hidden units, $\mathbf{J} \equiv \{\mathbf{J}_i\}_{1\leq i\leq K}$ is the set of input to hidden adaptive weights for the K hidden nodes and the hidden to output weights are set to one. The activation of hidden node i under presentation of the input pattern $\boldsymbol{\xi}^\mu$ is denoted $x_i^\mu = \mathbf{J}_i \cdot \boldsymbol{\xi}^\mu$. Training examples are input-output pairs of the form $(\boldsymbol{\xi}^\mu, \zeta^\mu)$ where μ labels each example in a sequence. The components of the independently drawn input vectors $\boldsymbol{\xi}^\mu$ are uncorrelated random variables with zero mean and unit variance. The corresponding output ζ^μ is given by a corrupted teacher of a similar configuration to the student except for a possible difference in the number M of hidden units: $\zeta^\mu = \sum_{n=1}^{M} g(\mathbf{B}_n \cdot \boldsymbol{\xi}^\mu) + \rho^\mu$, where $\mathbf{B} \equiv \{\mathbf{B}_n\}_{1\leq n\leq M}$ is the set of input to hidden adaptive weights for teacher hidden nodes and ρ^μ is zero-mean Gaussian output noise with variance σ^2. The activation of hidden node n under presentation of the input pattern $\boldsymbol{\xi}^\mu$ is denoted $y_n^\mu = \mathbf{B}_n \cdot \boldsymbol{\xi}^\mu$.

The error made by a student with weights $\mathbf{J}$ on a given input $\boldsymbol{\xi}$ is given by the quadratic deviation $\epsilon(\mathbf{J},\boldsymbol{\xi}) = \frac{1}{2}\left[\,\sigma(\mathbf{J},\boldsymbol{\xi}) - \zeta\,\right]^2$. The most basic learning rule is then to perform gradient descent on this quantity. Performance on a typical input in the absence of noise defines the generalization error $\epsilon_g(\mathbf{J}) \equiv \langle \epsilon(\mathbf{J},\boldsymbol{\xi})\rangle_{\{\xi\}}|_{\sigma=0}$ through an average over all possible input vectors $\boldsymbol{\xi}$.

The activations are distributed according to a multivariate Gaussian with covariances: $\langle x_i x_k\rangle = \mathbf{J}_i \cdot \mathbf{J}_k \equiv Q_{ik}$, $\langle x_i y_n\rangle = \mathbf{J}_i \cdot \mathbf{B}_n \equiv R_{in}$, and $\langle y_n y_m\rangle = \mathbf{B}_n \cdot \mathbf{B}_m \equiv T_{nm}$, measuring overlaps between student and teacher vectors. Angled brackets denote averages over inputs and the covariance matrix completely describes the mean state of the system. In the limit of large N we define a continuous time variable $\alpha = \mu/N$ and a coupled set of ordinary differential equations describes the overlap evolution under standard gradient descent [5]. These equations, representing an exact analytical solution for the average case, can be integrated numerically to obtain a solution of the dynamics. The generalization error can be written in terms of the overlaps and can thus be calculated once the dynamics have been solved. We will show that including an extra set of overlaps allows equations of motion to be determined for the matrix momentum learning algorithm.

3 Matrix momentum

A heuristic which is sometimes useful in batch learning is to include a momentum term in the basic gradient descent algorithm proportional to the previous change in the student's weights. For on-line momentum the weight update each iteration is given by,

$$\mathbf{J}_i^{\mu+1} = \mathbf{J}_i^\mu + \frac{\eta}{N}\delta_i^\mu \boldsymbol{\xi}^\mu + \beta(\mathbf{J}_i^\mu - \mathbf{J}_i^{\mu-1})\ , \qquad (1)$$

where $\delta_i^\mu \equiv g'(x_i^\mu)[\sum_{n=1}^{M} g(y_n^\mu) - \sum_{j=1}^{K} g(x_j^\mu) + \rho^\mu]$, β is a momentum parameter (which we consider to be scalar for now) and η is the learning rate which is scaled by the input dimension for convenience. Standard on-line momentum has been considered previously and has not been shown to be particularly useful (e.g. [4, 7, 8]) but it is instructive to consider this case first. We define a Markov process equivalent to equation (1) by introducing a new set of variables $\mathbf{\Omega}_i^\mu = N(\mathbf{J}_i^\mu - \mathbf{J}_i^{\mu-1})$,

$$\mathbf{J}_i^{\mu+1} = \mathbf{J}_i^\mu + \frac{\eta}{N}\delta_i^\mu \boldsymbol{\xi}^\mu + \frac{\beta}{N}\,\mathbf{\Omega}_i^\mu \;, \qquad \mathbf{\Omega}_i^{\mu+1} = \beta\,\mathbf{\Omega}_i^\mu + \eta\,\delta_i^\mu \boldsymbol{\xi}^\mu \;. \tag{2}$$

We can now proceed along the lines of [5] in order to derive a set of first order differential equations describing the evolution of a set of overlaps. In this case we need a new Gaussian field $z_i^\mu = \mathbf{\Omega}_i \cdot \boldsymbol{\xi}^\mu$ and a new set of overlaps: $\langle z_i z_k \rangle = \mathbf{\Omega}_i \cdot \mathbf{\Omega}_k \equiv C_{ik}$, $\langle z_i y_n \rangle = \mathbf{\Omega}_i \cdot \mathbf{B}_n \equiv D_{in}$, and $\langle x_i z_k \rangle = \mathbf{J}_i \cdot \mathbf{\Omega}_k \equiv E_{ik}$. We identify two possible scalings for η and β which result in different dynamical behaviour.

- If we choose $\eta \sim O(1)$ and $\beta \sim O(1/N)$ the above prescription results in an increasingly fast time scale for the new order parameters as N increases. This can be incorporated as an adiabatic elimination and we find that the dynamics of R and Q is simply equivalent to gradient descent with an effective learning rate of $\eta_{\text{eff}} = \eta/(1-\beta)$ in this case.
- More interesting dynamics is observed if we choose $\eta \sim O(1/N)$ and $1-\beta \sim O(1/N)$ [8]. In this case the overlaps all evolve on the same time-scale. If we define $\eta = k/N$ and $\beta = 1 - \gamma/N$ then taking $\gamma \to \infty$ and $k \to \infty$ simultaneously while keeping their ratio finite results in dynamics equivalent to gradient descent with an effective learning rate of $\eta_{\text{eff}} = k/\gamma$.

The above limits are related to those discussed in [7] and their results are consistent with the above observations. The latter scaling proves most appropriate for matrix momentum and is rigorously justified without resorting to adiabatic elimination. This is therefore the scaling discussed in the following two sections.

3.1 Idealized matrix momentum

Orr & Leen suggest the use of a matrix momentum parameter $\boldsymbol{\beta}$ so that the learning rate rescaling described in the previous section results in on-line Newton's method, in which the gradient is pre-multiplied by the inverse Hessian. If the Hessian is known this can be achieved by setting,

$$\boldsymbol{\beta} = \mathbf{I} - \frac{k\mathbf{H}}{N} \;, \qquad \eta = \frac{k\eta_\alpha}{N} \;, \tag{3}$$

where η_α is a scalar which may depend on α (recall that $\alpha = \mu/N$). Making k large one might then expect an effective matrix learning rate for gradient descent $\eta_{\text{eff}} = \eta_\alpha \mathbf{H}^{-1}$, as required for on-line Newton's method. Choosing $\eta_\alpha = 1/\alpha$ is known to provide optimal asymptotic performance in this case. However, it has not been shown that the limiting behaviour described for standard momentum holds for a matrix momentum parameter.

Substituting the above definitions into equations (2) and following the methods in [5] we find a set of differential equations for the overlaps as $N \to \infty$,

$$\frac{dQ_{ik}}{d\alpha} = E_{ik} + E_{ki} \;, \qquad \frac{dR_{in}}{d\alpha} = D_{in} \;,$$

$$\begin{aligned}
\frac{dC_{ik}}{d\alpha} &= k\eta_\alpha \langle \delta_i z_k + \delta_k z_i \rangle + k^2 \eta_\alpha^2 \langle \delta_i \delta_k \rangle - k \sum_m (c_{im} D_{km} + c_{km} D_{im}) \\
&\quad -k \sum_j (a_{ij} C_{kj} + a_{kj} C_{ij} + b_{ij} E_{jk} + b_{kj} E_{ji}) \ , \\
\frac{dD_{in}}{d\alpha} &= k\eta_\alpha \langle \delta_i y_n \rangle - k \sum_j (a_{ij} D_{jn} + b_{ij} R_{jn}) - k \sum_m c_{im} T_{nm} \ , \\
\frac{dE_{ik}}{d\alpha} &= C_{ik} + k\eta_\alpha \langle \delta_k x_i \rangle - k \sum_j (a_{kj} E_{ij} + b_{kj} Q_{ij}) - k \sum_m c_{km} R_{im} \ , \qquad (4)
\end{aligned}$$

where angled brackets denote averages over inputs, or equivalently averages over the field variables $\{x_i\}$, $\{y_n\}$ and $\{z_i\}$. We have defined,

$$\begin{aligned}
a_{ij} &= (1+\delta_{ij}) \frac{\partial \epsilon_g}{\partial Q_{ij}} \ , \\
b_{ij} &= (1+\delta_{ij}) \left[\sum_{lk} (1+\delta_{lk}) E_{lk} \frac{\partial^2 \epsilon_g}{\partial Q_{ij} \partial Q_{kl}} + \sum_{kn} D_{kn} \frac{\partial^2 \epsilon_g}{\partial Q_{ij} \partial R_{kn}} \right] \ , \\
c_{in} &= \sum_{lk} (1+\delta_{lk}) E_{lk} \frac{\partial^2 \epsilon_g}{\partial R_{in} \partial Q_{kl}} + \sum_{km} D_{km} \frac{\partial^2 \epsilon_g}{\partial R_{in} \partial R_{km}} \ ,
\end{aligned}$$

where δ_{ij} (with two indices) represents a Kronecker delta. The fields are distributed according to a multivariate Gaussian with the overlaps as covariances and all averages and generalization error derivatives can be calculated in closed form [5].

Matrix momentum, as defined above, is not particularly useful during the transients of learning in multilayer neural networks with over-lapping receptive fields, since both on-line Newton's method and matrix momentum can become trapped in a suboptimal fixed point of the dynamics [9]. This fixed point is an unstable transient fixed point for gradient descent [5] and it is therefore better to use gradient descent initially and to only switch on matrix momentum after escaping this fixed point. Other second order algorithms exist, such as natural gradient learning [2], which overcome this problem and provide improved transient performance over gradient descent [10]. Matrix momentum may also provide an efficient matrix inversion method for these algorithms.

In fig. 1(a) we compare the asymptotic performance of idealized matrix momentum to on-line Newton's method for a two-node network learning an isotropic task in the presence of output noise with $\sigma^2 = 0.01$ (The dynamics for on-line Newton's method is solved in [9]). We use gradient descent initially, until after the transient fixed point described above, and then we use matrix momentum with η_α gradually reduced from 0.1 to the $1/\alpha$ decay which is known to be asymptotically optimal. The dashed lines show results for $k = 0.01$, $k = 0.1$ and $k = 2$, in descending order of height (the final dashed line is almost obscured by the solid line). As k increases, the trajectory converges onto the on-line Newton's method result (solid line), as desired, and we approach the optimal asymptotic decay law (dot-dashed line) which is determined in [9]. Matrix momentum therefore provides an efficient approximation to on-line Newton's method when the Hessian is known.

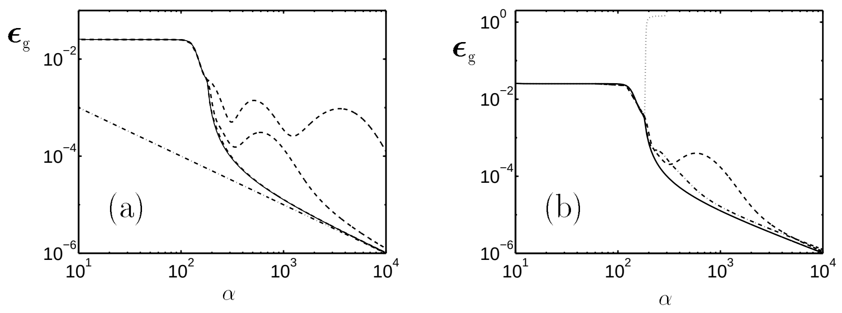

Figure 1: Solid lines show the generalization error for on-line Newton's method started after $\alpha = 180$, with gradient descent before this point, for a two hidden node network learning from a two node teacher ($T_{nm} = \delta_{nm}$, $\sigma^2 = 0.01$). (a) shows the result for idealized matrix momentum (dashed lines) for $k = 0.01$, $k = 0.1$ and $k = 2$ (in descending order of height). The dot-dashed line gives the optimal asymptotic decay. (b) shows the result for matrix momentum using the single example estimate with $k = 0.1$ (dashed), $k = 0.5$ (dot-dashed) and $k = 3$ (dotted). Initial conditions are $Q_{ii} \in U[0, 0.5]$, $Q_{i \neq k}, R_{in} \in U[0, 10^{-3}]$.

3.2 Single example approximation

In order to define a practical algorithm we need some approximation to the Hessian which can be determined on-line. The simplest such approximation is to use a single training example in order to estimate the Hessian [3]. The equations of motion for matrix momentum using this approximation can also be determined by the methods in [5] and the equations for Q and R are as in (4), while the equations for the other overlaps are,

$$\begin{aligned}
\frac{dC_{ik}}{d\alpha} &= k\langle(\eta_\alpha\delta_i - \phi_i)z_k + (\eta_\alpha\delta_k - \phi_k)z_i\rangle + k^2\langle(\eta_\alpha\delta_i - \phi_i)(\eta_\alpha\delta_k - \phi_k)\rangle \ , \\
\frac{dD_{in}}{d\alpha} &= k\langle(\eta_\alpha\delta_i - \phi_i)y_n\rangle \ , \qquad \frac{dE_{ik}}{d\alpha} = C_{ik} + k\langle(\eta_\alpha\delta_k - \phi_k)x_i\rangle \ . \qquad (5)
\end{aligned}$$

Again, the brackets denote averages over inputs, or fields, and we have defined $\phi_i = z_i g''(x_i) \left[\sum_j g(x_j) - \sum_n g(y_n) - \rho\right] + g'(x_i) \sum_j z_j g'(x_j)$. All averages can be carried out analytically to provide a closed set of equations of motion. In fig. 1(b) we show asymptotic performance with $k = 0.1$, $k = 0.5$ and $k = 3$ and η_α annealed as in fig. 1(a) after $\alpha = 180$. Ideally, we want the curves to approach the on-line Newton's method result (solid line) for large k. However, as k increases fluctuations in the Hessian estimate (due to randomness in the inputs and noise in the teacher output) become important and the weight vector norms diverge, leading to divergence of the generalization error (dotted line). For intermediate k (dot-dashed line) the performance is asymptotically close to optimal and certainly provides a significant improvement over gradient descent. Further work is required to determine the optimal and maximal values of k and η_α analytically, using methods from [6], but we have shown here that performance is certainly strongly dependent on parameter choice.

4 Conclusion and future work

In this paper we extend a recently developed theoretical framework to accommodate on-line second order methods and in particular we solve the dynamics of matrix momentum [3]. This algorithm provides a very efficient approximation to on-line Newton's method by avoiding explicit inversion of the Hessian and we show that the two methods are very close, if not equivalent, when the Hessian is known. The method is also reasonably stable to fluctuations caused by using a very crude single example approximation to the Hessian, as long as the algorithm parameters are chosen well. It should be reasonably straightforward to apply the results of [6] in order to determine optimal and maximal parameters in this case, in terms of task complexity and non-linearity. However, to obtain more robust performance a better on-line approximation to the Hessian should probably be used.

Hessian based methods are not appropriate during the transients of learning because there is a possibility of trapping in suboptimal fixed points [9]. It would therefore make sense to use other matrix pre-multipliers which are guaranteed positive definite, such as the Fisher information matrix used in natural gradient learning [2, 10] or the linearized Hessian [4] used in Gauss-Newton methods. Matrix momentum could easily provide an efficient inversion method in order to approximate the resulting algorithms.

Acknowledgement This work was supported by the EPSRC grant GR/L19232.

References

[1] V. Fabian. *Ann. Math. Statist.*, **39**, 1327 (1968).

[2] S. Amari *Neural Computation* **10**(2) 251 (1998).

[3] G. B. Orr, T. K. Leen *Advances in Neural Information Processing Systems* vol 9, ed M. C. Mozer, M. I. Jordan and T. Petsche (Cambridge, MA: MIT Press, 1997) p 606

[4] G. B. Orr, Ph.D. Dissertation, Oregon Graduate Institute of Science & Technology (1995).

[5] D. Saad, S. A. Solla, *Phys. Rev. Lett.* **74**, 4337 (1995); *Phys. Rev. E* **52** 4225 (1995).

[6] T. K. Leen, B. Schottky, D. Saad *Advances in Neural Information Processing Systems* vol 10, ed M. I. Jordan, M. J. Kearns and S. A. Solla (Cambridge, MA: MIT Press, 1998).

[7] W. Weigerinck, A. Komoda, T .Heskes *J. Phys. A* **27**, 4425 (1994).

[8] A. Prügel-Bennett, unpublished notes, (1996).

[9] M. Rattray, D. Saad, 'Incorporating curvature information into on-line learning' *Proc. of the On-line Learning Themed Week*, (Isaac Newton Institute, Cambridge, 1997).

[10] M. Rattray, D. Saad, S. A. Solla, S. Amari 'Natural gradient descent for on-line learning' (in preparation, 1998).