Incorporating Curvature Information into On-line Learning

Magnus Rattray † *and David Saad* ‡

Neural Computing Research Group, Aston University
Birmingham B4 7ET, UK.
†*rattray@aston.ac.uk*
‡*saadd@aston.ac.uk*

Abstract

We analyse the dynamics of a number of second order on-line learning algorithms training multi-layer neural networks, using the methods of statistical mechanics. We first consider on-line Newton's method, which is known to provide optimal asymptotic performance. We determine the asymptotic generalization error decay for a soft committee machine, which is shown to compare favourably with the result for standard gradient descent. Matrix momentum provides a practical approximation to this method by allowing an efficient inversion of the Hessian. We consider an idealized matrix momentum algorithm which requires access to the Hessian and find close correspondence with the dynamics of on-line Newton's method. In practice, the Hessian will not be known on-line and we therefore consider matrix momentum using a single example approximation to the Hessian. In this case good asymptotic performance may still be achieved, but the algorithm is now sensitive to parameter choice because of noise in the Hessian estimate. On-line Newton's method is not appropriate during the transient learning phase, since a suboptimal unstable fixed point of the gradient descent dynamics becomes stable for this algorithm. A principled alternative is to use Amari's natural gradient learning algorithm and we show how this method provides a significant reduction in learning time when compared to gradient descent, while retaining the asymptotic performance of on-line Newton's method.

1 Introduction

On-line learning is a popular method for training multi-layer feed-forward neural networks, especially for large systems and for problems requiring rapid and adaptive data processing. Under the on-line learning framework, network parameters are updated according to only the latest in a sequence of training examples. This is to be contrasted with batch methods which utilise the entire

training set at each learning iteration. On-line methods can be beneficial in terms of both storage and computation time, and also allow for temporal changes in the task being learned.

The most basic on-line learning algorithm for models which are differentiable with respect to their parameters is stochastic gradient descent. Given some differentiable error function, the network weights are adapted in the negative gradient direction of this error calculated according to only the current, randomly drawn, training example. Under the batch learning framework (and in other optimization problems) it is well known that curvature information can be used in order to speed up learning (see, for example, Bishop, 1995). Typically this curvature information is in the form of some estimate of the Hessian matrix or its inverse, as required for Newton-type algorithms (unless otherwise stated, we define the Hessian as the matrix of second derivatives of the error averaged over the entire training set). Pre-multiplying the standard gradient with the inverse Hessian and annealing the learning rate appropriately provides asymptotically optimal performance when emulating stochastic rules, equalling even the best batch algorithm (Amari, 1998). However, determining the Hessian on-line is difficult as we only have access to a single training example at any one time. Even if the Hessian can be estimated on-line, inverting it will be computationally costly when our network is large. This is particularly undesirable when we consider that computational efficiency is one of the principle reasons for using on-line methods.

Despite these difficulties a number of algorithms have been proposed which estimate curvature information on-line. For example, Le Cun *et al* (1993) describe an on-line method for determining eigenvalues of the Hessian, which allows an appropriate learning rate to be used for gradient descent at late times. Orr & Leen (1994, 1997) have recently introduced an on-line matrix momentum algorithm in order to invert an estimate of the Hessian on-line. This latter method is particularly interesting since the inversion is replaced by a matrix-vector multiplication which can be carried out by an efficient back-propagation step. On-line versions of other second order methods are also available (for a review, see Bishop, 1995).

A different approach has recently been proposed by Amari (1998), who has introduced a natural gradient learning algorithm inspired by ideas from information geometry. When learning to emulate a stochastic rule with some probabilistic model this learning algorithm has the desirable properties of asymptotic optimality (for a sufficiently rich model) and invariance to reparameterizations of our model distribution. This latter property is achieved by viewing the parameter space of the model as a Reimannian space in which local distance is defined by the Kullback-Leibler divergence. This method requires knowledge of the input distribution and the inversion of a large matrix (the Fisher information) but in some cases the algorithm can be executed with relatively low cost (Yang & Amari, 1998). The natural gradient method is intended to provide improved performance during both transient

and asymptotic stages of learning and we will see that this is certainly true for the examples presented here.

In this paper we model the dynamics displayed by some of the above learning algorithms using a recently developed statistical mechanics framework. This framework allows accurate modelling for on-line learning in two-layer networks with large input dimension and provides a compact and easily interpretable description of the learning process (Biehl & Schwarze, 1995; Saad & Solla, 1995).

We first solve the dynamics for an idealized on-line version of Newton's method which uses knowledge of the exact Hessian. In this case we show how unstable transient fixed points, which can appear in gradient descent training of multi-layer networks with over-lapping receptive fields, can become attractive fixed points in this case. This highlights a significant limitation for Hessian based algorithms, which is easily explained by examining the behaviour of the algorithm close to the fixed point. As expected, asymptotic performance is shown to be significantly better than for standard gradient descent and we provide some generic asymptotic results in terms of task complexity and non-linearity.

As we have already said, the true Hessian will not be known in general and must somehow be estimated if we wish to obtain optimal asymptotic performance. An efficient inversion method is also required. We therefore consider Orr & Leen's matrix momentum algorithm: firstly we show how the inversion is achieved for an idealized algorithm in which the true Hessian is known and secondly we examine the efficacy of using a rather crude on-line estimate of the Hessian.

Matrix momentum still suffers from the problem that previously transient fixed points become stable. By using an alternative matrix pre-multiplier which is guaranteed positive definite one can avoid this problem. A number of possibilities exist, yet these alternatives do not really have any principled justification outside the asymptotic regime (for example, Orr (1995) uses the linearized Hessian, as do Gauss-Newton methods). A more principled choice is to use natural gradient learning (Amari, 1998). We model the dynamics of this algorithm and show how performance is much improved over standard gradient descent in all phases of learning.

2 Statistical mechanics framework

A statistical mechanics framework is used to obtain a compact description of the learning dynamics, which is exact for large input dimension N and provides an accurate model of mean behaviour for realistic N (Biehl & Schwarze, 1995; Saad & Solla, 1995; Barber *et al*, 1996). We consider a mapping 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 defines a soft committee machine, where we

choose activation function $g(x) \equiv \mathrm{erf}(x/\sqrt{2})$, $\mathbf{J} \equiv \{\mathbf{J}_i\}_{1\leq i\leq K}$ 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 $x_i^\mu = \mathbf{J}_i \cdot \boldsymbol{\xi}^\mu$. This configuration preserves most properties of general multi-layer networks and can be extended to accommodate adaptive hidden to output weights (Reigler & Biehl, 1995).

Training examples are of the form $(\boldsymbol{\xi}^\mu, \zeta^\mu)$ where μ labels each example and components of the independently drawn input vectors $\boldsymbol{\xi}^\mu$ are uncorrelated and come from a Gaussian distribution 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: $\zeta^\mu = \sum_{n=1}^{M} g\left(\mathbf{B}_n \cdot \boldsymbol{\xi}^\mu\right) + \rho^\mu$, where $\mathbf{B} \equiv \{\mathbf{B}_n\}_{1\leq n\leq M}$ is the set of input to hidden adaptive weights and ρ^μ is Gaussian output noise with variance σ^2. The activation of hidden node n under presentation of the input pattern $\boldsymbol{\xi}^\mu$ is denoted $y_n^\mu = \mathbf{B}_n \cdot \boldsymbol{\xi}^\mu$. Where possible, we will use indices $i, j, k, l \ldots$ to refer to units in the student network and $n, m, \ldots$ for units in the teacher network.

The error made by a student with weights $\mathbf{J}$ on a given input $\boldsymbol{\xi}$ is given by the quadratic deviation

$$\epsilon_{\mathbf{J}}(\boldsymbol{\xi}, \zeta) = \frac{1}{2}[\ \sigma(\mathbf{J}, \boldsymbol{\xi}) - \zeta\ ]^2 = \frac{1}{2}\left[\ \sum_{i=1}^{K} g(x_i) - \sum_{n=1}^{M} g(y_n) - \rho\ \right]^2 \quad , \tag{2.1}$$

which is proportional to the log-likelihood of the data under a Gaussian noise model. Performance on a typical input in the absence of noise defines the generalization error $\epsilon_g(\mathbf{J}) \equiv \langle \epsilon_{\mathbf{J}}(\boldsymbol{\xi}, \zeta)\rangle_{\{\xi\}}|_{\sigma=0}$ 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. The covariance matrix completely describes the state of the system, in the limit of large N, enabling us to write down a closed set of ordinary differential equations for the evolution of each one of the overlaps under standard gradient descent (Saad & Solla, 1995). In addition, the generalization error may be written exclusively in terms of the overlaps so that these equations of motion are sufficient to describe the evolution of the generalization error. The equations, representing an exact analytical solution for the average case, can be integrated numerically to obtain a solution of the dynamics. In the following sections we will show how this framework can be generalized to describe the dynamics for a number of second order learning algorithms.

3 On-line Newton's method

The Hessian cannot be determined on-line in practice and on-line Newton's method is therefore mainly of theoretical interest. However, we consider this idealized algorithm here so that we can better understand algorithms like matrix momentum, which seek to emulate the performance of on-line Newton's method.

3.1 The Hessian

We define the Hessian to be the matrix of second derivatives of the training error with respect to the weights, averaged over all training examples. We will consider an unlimited number of examples, in which case this is simply the second derivative of the generalization error. The Hessian is made up of K^2 blocks $\mathbf{H} = [\mathbf{H}_{ik}]$ which can be determined as described in appendix A,

$$\begin{aligned}\mathbf{H}_{ik} = \mathbf{I}\,(1+\delta_{ik})\frac{\partial \epsilon_g}{\partial Q_{ik}} + \sum_{jl} \mathbf{J}_j \mathbf{J}_l^{\mathrm{T}} (1+\delta_{ij})(1+\delta_{kl}) \frac{\partial^2 \epsilon_g}{\partial Q_{ij} \partial Q_{kl}} \\ + \sum_{jn} \mathbf{J}_j \mathbf{B}_n^{\mathrm{T}} (1+\delta_{ij}) \frac{\partial^2 \epsilon_g}{\partial Q_{ij} \partial R_{kn}} + \sum_{nj} \mathbf{B}_n \mathbf{J}_j^{\mathrm{T}} (1+\delta_{jk}) \frac{\partial^2 \epsilon_g}{\partial R_{in} \partial Q_{jk}} \\ + \sum_{nm} \mathbf{B}_n \mathbf{B}_m^{\mathrm{T}} \frac{\partial^2 \epsilon_g}{\partial R_{in} \partial R_{km}} \,. \end{aligned} \tag{3.1}$$

The generalization error can be written in closed form as a function of Q, R and T (Saad & Solla, 1995) in which case the above expression is also in closed form (the generalization error and derivatives are given in appendix B). Each block of the Hessian takes the form of an identity matrix added to outer products of weight vectors and it is straightforward to show that each block of the inverse will also be of this general form. Inversion can be carried out by partitioning and we show how to calculate the inverse for $K = 2$ in appendix A.1 (this result can easily be generalized to larger K). Each block of the inverse Hessian can then be written in the following form,

$$\mathbf{H}_{ik}^{-1} = \alpha_{ik}(\mathbf{I} + \mathbf{S}\Theta^{ik}\mathbf{S}^{\mathrm{T}}) \,, \tag{3.2}$$

where $\mathbf{S} = (\mathbf{J}_1, \ldots, \mathbf{J}_K, \mathbf{B}_1, \ldots, \mathbf{B}_M)$, α_{ik} is a scalar coefficient and Θ^{ik} is an $M+K$ dimensional square matrix (these only depend on the order parameters Q, R and T).

3.2 Equations of motion

Given the inverse Hessian, an on-line version of Newton's method is defined by the following weight update at each iteration,

$$\mathbf{J}^{\mu+1} = \mathbf{J}^{\mu} - \frac{\eta}{N} \mathbf{H}^{-1} \nabla_{\mathbf{J}}\, \epsilon_{\mathbf{J}}(\boldsymbol{\xi}^{\mu}, \zeta) \,, \tag{3.3}$$

where the learning rate η has been scaled with the input size N and may depend on α in general. The weights to hidden node i are then updated as follows,

$$\mathbf{J}_i^{\mu+1} = \mathbf{J}_i^{\mu} + \frac{\eta}{N} \sum_{j=1}^{K} \mathbf{H}_{ij}^{-1} \delta_j^{\mu} \boldsymbol{\xi}^{\mu} \ , \tag{3.4}$$

where $\delta_i^{\mu} \equiv g'(x_i^{\mu})[\sum_{n=1}^{M} g(y_n^{\mu}) - \sum_{j=1}^{K} g(x_j^{\mu}) + \rho^{\mu}]$.

In the large N limit the order parameters change according to a completely deterministic trajectory. Using the methods developed in (Saad & Solla, 1995) it is straightforward to write down a coupled set of differential equations describing this trajectory,

$$\begin{aligned} \frac{\mathrm{d}R_{in}}{\mathrm{d}\alpha} &= \eta\, f_{in}(R,Q,T) \ , \\ \frac{\mathrm{d}Q_{ik}}{\mathrm{d}\alpha} &= \eta\, g_{ik}(R,Q,T) + \eta^2 h_{ik}(R,Q,T,\sigma^2) \ , \end{aligned} \tag{3.5}$$

where we have defined a new time variable $\alpha = \mu/N$ to be the normalized number of patterns presented so far. This is the same general form as for standard gradient descent (explicit expressions for f, g and h are given in appendix C.1). All the effects of stochasticity are contained within the η^2 term, which is analogous to a "diffusion" term in the language of stochastic dynamics. This term is proportional to the variance of the input distribution and contains an additive contribution proportional to the noise variance. The terms linear in η contribute effects due to motion on the mean error (generalization error) surface. Although it is these "drift" terms which decide most qualitative features of the learning dynamics outside the asymptotic regime, the diffusion term limits increases in the learning rate and is of key importance for determining the appropriate maximal and optimal learning parameters (West & Saad, 1997; Saad & Rattray, 1997, this volume).

3.3 Integrating the dynamics

The learning dynamics for gradient descent have been well studied and we give a brief description of the main features here, before discussing on-line Newton's method (Saad & Solla, 1995; Saad & Rattray, this volume). The order parameters are initialized randomly, with overlaps between different vectors (R_{in} and $Q_{i\neq k}$) taking small values of $O(1/\sqrt{N})$ while the student norms Q_{ii} are $O(1)$. We choose values corresponding to $N \sim 10^6$ but different initial conditions just lead to changes in the learning time-scale, with learning times growing logarithmically with N (Biehl *et al*, 1996). The overall shape of the learning curve and the optimal and maximal learning rates in each phase are not affected by the choice of initial conditions. Finite size effects have been studied by Barber *et al* (1995), showing that the picture described here holds for much smaller N.

As gradient descent learning begins, the order parameters quickly converge to a transient fixed point, the symmetric phase, which is characterized by close similarity of all student-teacher overlaps. This fixed point is unstable, however, and small differences in the initial conditions diverge exponentially (hence the logarithmic scaling of learning time with N). Once the student-teacher overlaps diverge sufficiently the system leaves this transient fixed point and the order parameters converge towards their asymptotic values (or possibly to another transient fixed point first). There will always be at least one transient fixed point as long as $K \geq 2$ (unless the learning rate is chosen too large for successful learning), even if the teacher is a perceptron. For matched teacher and student ($M = K$) $R_{in} \to T_{in}$ and $Q_{ik} \to T_{ik}$ asymptotically (for appropriately ordered indices) as long as the learning rate is chosen well. In the absence of noise and for $K \geq M$ asymptotic convergence is exponential as long as the learning rate is fixed and not too large. For unrealizable learning (noisy and/or with $K < M$) the learning rate should be annealed inversely with α for optimal asymptotic performance (Leen *et al*, 1997) in which case the generalization error also converges according to an inverse power law (if the prefactor for the learning rate decay is not too small).

In fig. 1(a) we show the evolution of the generalization error under gradient descent for various learning rates, for a noiseless, architecturally matched learning scenario ($K = M$) and an isotropic teacher ($T_{nm} = \delta_{nm}$). A scaled learning rate $\tilde{\alpha} = \alpha\eta$ is used, which reflects the scaling invariance of equations (3.5) for small η and allows us to meaningfully take the $\eta \to 0$ limit. The plateau in generalization error is due to the symmetric phase described above and dominates the learning time.

In fig. 1(b) we plot the corresponding results for on-line Newton's method. The system never leaves the symmetric phase in this case and it appears that the symmetric fixed point is now stable to perturbations. We have not found any situation for large or small η in which the system leaves this fixed point (for a variety of learning scenarios). It is easy to see why on-line Newton's method will make any fixed point stable when the Hessian is non-singular (this is exactly the effect of Newton's method, which diagonalizes the linearized dynamics). However, the symmetric fixed point is characterized by a singular Hessian, in which case this simple picture no longer holds. We have studied the $K = M = 2$ case analytically for small learning rate, in a similar analysis to that of Saad & Solla (1995) for gradient descent. In this limit the student weight vectors are assumed to lie in a subspace spanned by the teacher weight vectors. The system is then completely determined by the student-teacher overlaps. Exploiting symmetries observed in the dynamics ($R_{in} = R\delta_{in} + S(1 - \delta_{in})$) we have a further simplified two dimensional system, which we study by a linear expansion around the symmetric fixed point. For gradient descent a single positive eigenvalue results in the eventual divergence of R and S. The picture is different for on-line Newton's method however and here we find that both eigenvalues are negative, resulting in a stable fixed point. This

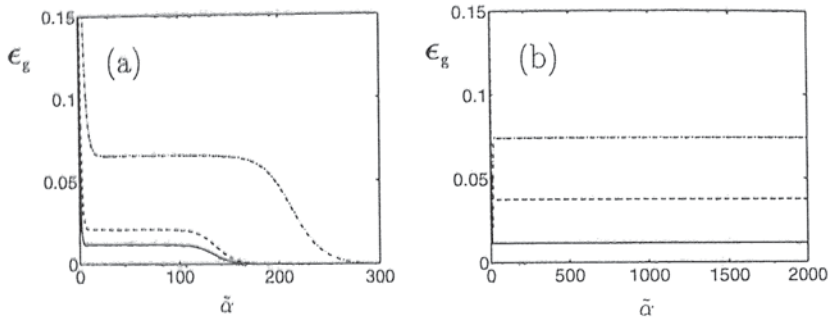

Fig. 1: We compare the performance of gradient descent (a) and on-line Newton's method (b) for a two hidden node network learning from examples generated by a two node isotropic teacher ($T_{nm} = \delta_{nm}$) in the absence of noise. Curves show the generalization error for learning rates $\eta = 2$ (dash-dotted line), $\eta = 1$ (dashed line) and $\eta \to 0$ (solid line) against a scaled time variable $\tilde{\alpha} = \alpha\eta$. Initial conditions are $Q_{ii} \in U[0, 0.5]$, $Q_{i \neq k}, R_{in} \in U[0, 10^{-3}]$.

analysis exemplifies the usefulness of the order parameters approach, since a stochastic approximation analysis would be made difficult by the singular Hessian at this fixed point (this is equally true for standard gradient descent).

It is not guaranteed that diffusion terms will not make this fixed point unstable, although in our case this does not appear to be the case (it is difficult to study the finite η situation analytically since the fixed point can no longer be determined exactly in this case). This corresponds with what one might expect, given that diffusion does not aid escape from the symmetric phase for standard gradient descent (West & Saad, 1997). We note here that the inclusion of noise would require a consideration of second order terms (as in the asymptotic annealing analysis described below), but for low noise levels these terms will only become relevant within the immediate neighbourhood of a slightly shifted fixed point.

3.4 Asymptotic performance

In the previous section we found that on-line Newton's method is susceptible to trapping in a transient, suboptimal fixed point. However, in the presence of noise, optimal asymptotic performance will be achieved if on-line Newton's method is used at late times with learning rate $1/\alpha$. The asymptotic dynamics for gradient descent with an annealed learning rate has recently been solved

under the statistical mechanics formalism and the optimal generalization error decay is known in this case (Leen *et al*, 1997). Here we extend those results to on-line Newton's method.

Asymptotically for an isotropic realizable task ($K = M$, $T_{nm} = T\delta_{nm}$) we can examine a four dimensional system by defining $R_{in} = R\delta_{in} + S(1 - \delta_{in})$ and $Q_{ik} = Q\delta_{ik} + C(1 - \delta_{ik})$, as this avoids degeneracy in the dynamical equations (the initial conditions, which would break this symmetry, become negligible asymptotically). This approach allows general results in terms of K and T. We define $\mathbf{u} = (R - T, Q - T, S, C)^{\mathrm{T}}$ to be the deviation from the asymptotic fixed point. If the learning rate decays according to some power law then the linearized equations of motion around this fixed point are given by,

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\alpha} = \eta \mathbf{M}\,\mathbf{u} + \eta^2\sigma^2\mathbf{b}\ , \tag{3.6}$$

where $\eta\mathbf{M}$ is the Jacobian of the equations of motion to first order in η while the only non-vanishing second order terms are proportional to the noise variance. The asymptotic equations of motion can be determined using the asymptotic expression for the inverse Hessian (see appendix A.4). A more detailed account will be provided elsewhere (Rattray & Saad, 1998) and here we just provide the solution to the above equation with $\eta = \eta_0/\alpha$,

$$\mathbf{u}(\alpha) = \sigma^2\mathbf{V}\mathbf{X}\mathbf{V}^{-1}\mathbf{b}\ , \tag{3.7}$$

where $\mathbf{V}^{-1}\mathbf{M}\mathbf{V}$ is a diagonal matrix whose entries λ_i are eigenvalues of $\mathbf{M}$ and we have defined the diagonal matrix $\mathbf{X}$ to be,

$$\mathbf{X}_i^{\mathrm{diag}} = -\frac{\eta_0^2}{1+\lambda_i\eta_0}\left[\frac{1}{\alpha} - \alpha^{\lambda_i\eta_0}\alpha_0^{-(1+\lambda_i\eta_0)}\right]\ , \tag{3.8}$$

with annealing beginning at $\alpha = \alpha_0$. We find two degenerate eigenvalues $\lambda_{1,2} = -1$, $\lambda_{3,4} = -2$ and by substituting equation (3.8) into a first order expansion of the generalization error it is straightforward to show $\eta_0 = 1$ to be optimal, as expected. In this case the modes corresponding to $\lambda_{1,2}$ do not contribute to the asymptotic generalization error and we find a particularly simple decay law which is independent of T,

$$\epsilon_g = \frac{\sigma^2 K}{2\alpha}\ . \tag{3.9}$$

In fig. 2 we compare the prefactor of the optimal generalization decay ($\epsilon_g \sim \epsilon_0\sigma^2/\alpha$) for on-line Newton's method with the gradient descent results from (Leen *et al*, 1997). The result for gradient descent is not exactly linear in K, but quickly approaches a linear scaling as K increases (see fig. 2(a)). In fig. 2(b) we show how performance differs most when T becomes small, while the optimal gradient descent decay approaches the result for on-line Newton's

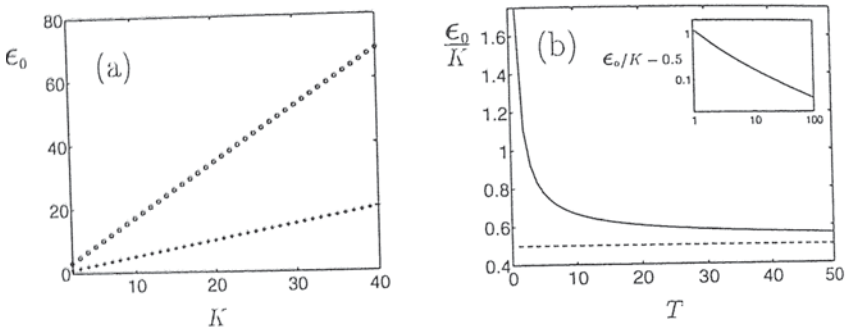

Fig. 2: Prefactor for the asymptotic decay of the generalization error ($\epsilon_g \sim \epsilon_0 \sigma^2/\alpha$): (a) shows the prefactor for $T = 1$ as a function of K for optimal gradient descent (circles) and on-line Newton's method (crosses) while (b) shows how the prefactor for optimal gradient descent (large K) decays towards 0.5 as T increases, which is the prefactor for on-line Newton's method. Adapted from (Rattray *et al*, 1998).

method for large T. This can be explained by examining the asymptotic expression for the Hessian, shown in equation (A.13). For large T the diagonals of the Hessian are $O(1/\sqrt{T})$ and equal (for large N) while all other terms are at most $O(1/T)$, so that the Hessian is proportional to the identity matrix in this limit and Newton's method is effectively equivalent to gradient descent. However, for small T the diagonals are $O(T^2)$ while the off-diagonals remain finite, so that the Hessian is dominated by off-diagonals in this limit.

Although the optimal learning rate decay for gradient descent is inversely proportional to α, the prefactor is strongly problem dependent. This is not so for on-line Newton's method, for which $\eta = 1/\alpha$ is always optimal. We also note that if the prefactor is chosen too small in gradient descent, the generalization error will follow a slower power law decay.

In fig. 3 the approach to the asymptotic decay is shown for an example of realizable learning ($K = M = 2$) with an isotropic teacher and noise variance $\sigma^2 = 0.01$. The optimal decay law is shown by the dot-dashed line in fig. 3(a) while the solid line gives the generalization error for on-line Newton's method, initialized after the symmetric phase at $\alpha_i = 180$ (before this point gradient descent is used with $\eta = 1$). The learning rate is annealed from some appropriate constant (we choose $\eta_i = 0.1$) according to the following prescription,

$$\eta = \frac{\eta_i}{1 + (\alpha - \alpha_i)\eta_i} \,. \tag{3.10}$$

We see how losses incurred due to trapping during the symmetric phase result in a rather late approach to the final, optimal decay. If the learning rate is simply chosen equal to $1/\alpha$ then the approach is much slower.

4 Matrix momentum

A heuristic which is sometimes useful in batch learning is to include a momentum term in the basic gradient descent algorithm (for a discussion, see Bishop, 1995). For on-line learning with momentum we have,

$$\mathrm{J}_i^{\mu+1} = \mathrm{J}_i^{\mu} + \frac{\eta}{N}\delta_i^{\mu}\boldsymbol{\xi}^{\mu} + \beta(\mathrm{J}_i^{\mu} - \mathrm{J}_i^{\mu-1}) \;, \tag{4.1}$$

where $\delta_i^{\mu} \equiv g'(x_i^{\mu})[\sum_{n=1}^{M} g(y_n^{\mu}) - \sum_{j=1}^{K} g(x_j^{\mu}) + \rho^{\mu}]$. This is the same as for standard gradient descent except for the inclusion of a term proportional to the previous weight update.

On-line momentum has been considered previously but has not been found to be particularly useful, except perhaps in smoothing the asymptotic trajectory of the weights (Roy & Shynk, 1990; Weigerinck *et al*, 1994; Orr, 1995). However, by choosing an appropriate matrix momentum parameter one may obtain close to optimal asymptotic performance (Orr & Leen, 1994, 1997). Before introducing matrix momentum it will be useful to consider the dynamics of standard momentum for large N.

4.1 Standard momentum

Equation (4.1) defines a second order process, in which weights from the two previous iterations are required for each update. We define an equivalent first order process by introducing a new set of variables $\Omega_i^{\mu} = N(\mathrm{J}_i^{\mu} - \mathrm{J}_i^{\mu-1})$,

$$\begin{aligned} \mathrm{J}_i^{\mu+1} &= \mathrm{J}_i^{\mu} + \frac{\eta}{N}\delta_i^{\mu}\boldsymbol{\xi}^{\mu} + \frac{\beta}{N}\,\Omega_i^{\mu} \;, \\ \Omega_i^{\mu+1} &= \beta\,\Omega_i^{\mu} + \eta\,\delta_i^{\mu}\boldsymbol{\xi}^{\mu} \;. \end{aligned} \tag{4.2}$$

We can now proceed along the lines of (Saad & Solla, 1995) in order to derive a set of first order differential equations describing the evolution of a set of order parameters. In this case we need a new Gaussian field $z_i^{\mu} = \Omega_i \cdot \boldsymbol{\xi}^{\mu}$ and a new set of order parameters: $\langle z_i z_k \rangle = \Omega_i \cdot \Omega_k \equiv C_{ik}$, $\langle z_i y_n \rangle = \Omega_i \cdot \mathrm{B}_n \equiv D_{in}$, and $\langle x_i z_k \rangle = \mathrm{J}_i \cdot \Omega_k \equiv E_{ik}$. We identify two possible scaling 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 $\eta_{\mathrm{eff}} = \eta/(1-\beta)$ in this case.

- More interesting dynamics is observed if we choose $\eta \sim O(1/N)$ and $1-\beta \sim O(1/N)$ (Prügel-Bennett, 1997). In this case the order parameters 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 $\eta_{\rm eff} = k/\gamma$.

The above limits are related to those discussed by Weigerinck *et al* (1994) 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 sections.

4.2 Idealized matrix momentum

Orr & Leen suggest the use of a matrix momentum parameter β so that the learning rate rescaling described in the previous section results in on-line Newton's method. If the Hessian is known this can be achieved by setting,

$$\beta = \mathbf{I} - \frac{k\mathbf{H}}{N} , \qquad \eta = \frac{k\eta_\alpha}{N} , \tag{4.3}$$

where η_α is a scalar which may depend on α. Making k large one might then expect an effective matrix learning rate,

$$\eta_{\rm eff} = \eta_\alpha \mathbf{H}^{-1} , \tag{4.4}$$

as required for on-line Newton's method. However, there are two problems with this result: it has not been shown that the limiting behaviour described for standard momentum holds for a matrix momentum parameter and we do not have on-line access the Hessian. In this section we address the first issue by solving the matrix momentum dynamics for an idealized situation in which the Hessian is known. In the following section we consider an approximation based on using only the latest training example to estimate the Hessian.

Substituting the above definitions into equations (4.2) using the definition of the Hessian given in equation (3.1) and following the methods of Saad & Solla (1995) we find a coupled set of differential equations for the order parameters as $N \to \infty$, which are given in appendix C.2.

In fig. 3(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 noise ($\sigma^2 = 0.01$). Both methods become trapped in the symmetric fixed point, as explained in the previous section, so we use gradient descent initially and after the symmetric phase we use matrix momentum with η_α annealed according to equation (3.10). 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). Matrix momentum therefore provides an efficient approximation to on-line Newton's method when the Hessian is known. In the next section we consider a realizable algorithm which uses an approximation to the Hessian.

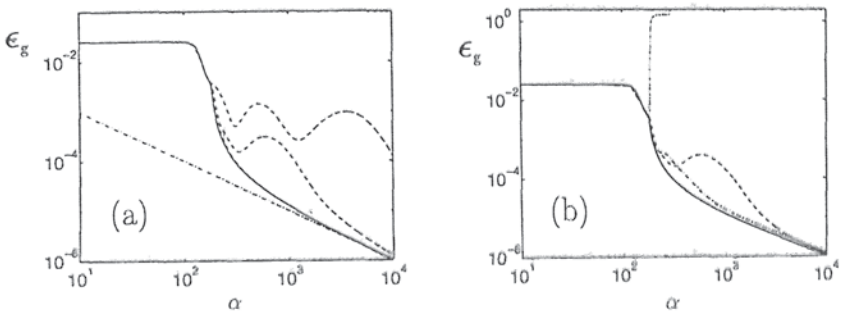

Fig. 3: The solid lines in (a) and (b) show the generalization error for annealed on-line Newton's method started after $\alpha = 180$, with gradient descent before this point, for a two hidden node network learning from examples generated by a two node isotropic teacher ($T_{nm} = \delta_{nm}$) corrupted by noise ($\sigma^2 = 0.01$). In (a) we show the corresponding generalization error 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. In (b) we show the generalization error for matrix momentum using a single pattern estimate for the Hessian with $k = 0.1$ (dashed), $k = 0.5$ (dot-dashed) and $k = 3$ (dotted). Initial conditions are as in fig. 1 (order parameters specific to matrix momentum are initialized to zero).

4.3 Single pattern 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 (Orr & Leen, 1997). The single-pattern Hessian is written in appendix A.2 and the equations of motion for matrix momentum using this approximation are given in appendix C.3.

In figure 3(b) we show the asymptotic performance of matrix momentum using the single pattern approximation, for a two node network learning an isotropic task in the presence of noise ($\sigma^2 = 0.01$). Curves are shown for $k = 0.1$, $k = 0.5$ and $k = 3$, with η_α chosen according to equation (3.10) after $\alpha = 180$. Ideally, we would wish for 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) 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 (Leen *et al*, 1997), but we have shown here that performance is certainly strongly dependent on parameter choice. It would be interesting to consider more sophisticated on-line approximations to the Hessian, which might provide greater robustness.

5 Natural gradient learning

As we saw in the section 3, on-line Newton's method does not guarantee convergence to a minimum of the generalization error because the Hessian is not always positive definite. A number of heuristics exist which ensure the matrix pre-multiplier of the gradient is positive definite; for example Orr (1995) suggests using the linearized Hessian (as in the Gauss-Newton method), or one could add the identity matrix multiplied by some scalar parameter to the Hessian (or its inverse), with the parameter reduced to zero asymptotically. These methods do guarantee asymptotic optimality and convergence to a minimum of the generalization error, but lack any principled motivation during the transient phases of learning.

A more principled approach has recently been proposed by Amari (1998). Natural gradient learning ensures asymptotic optimality, given a sufficiently rich model, is invariant to reparameterization of our model distribution (defined by the student in our case) and always converges to a local minimum of the generalization error if the learning rate is annealed appropriately. Invariance to reparameterization is achieved by viewing the parameter (weight) space of the model as a Reimannian space in which local distance is defined by the Kullback-Leibler divergence (Yang & Amari, 1997). The Fisher information matrix then plays the role of a Reimannian metric in this space. The natural gradient learning rule is obtained by pre-multiplying the gradient of the log-likelihood (of the most recent training example) with the inverse of this matrix, which plays a similar role to the Hessian in on-line Newton's method.

5.1 Fisher Information Matrix

Our model distribution is taken to be the student network with output corrupted by zero mean Gaussian noise of variance $\sigma_{\rm m}^2$. The error defined by equation (2.1) is proportional to the log-likelihood of the latest training example under this noise model. Each entry in the Fisher information matrix $\mathrm{G} = [G_{i\alpha,k\beta}]$, where $1 \leq i,k \leq K$ and $1 \leq \alpha,\beta \leq N$, is defined,

$$G_{i\alpha,k\beta} = \frac{1}{\sigma_{\rm m}^4} \left\langle \frac{\partial \epsilon_{\rm J}(\xi,\zeta_{\rm J})}{\partial J_{i\alpha}} \frac{\partial \epsilon_{\rm J}(\xi,\zeta_{\rm J})}{\partial J_{k\beta}} \Big|_{\zeta_{\rm J}=\sum_i g(x_i)+\rho_{\rm m}} \right\rangle_{\{\rho_{\rm m},\xi\}} . \tag{5.1}$$

Here, the brackets denote an average over the input distribution and model noise $\rho_{\rm m}$, which is taken from a Gaussian distribution with zero mean and variance $\sigma_{\rm m}^2$. Amari (1998) has determined the Fisher information matrix for a general two-layer network and for our particular choice of activation function with a Gaussian input distribution we find $\mathrm{G} = \mathrm{A}/\sigma_{\rm m}^2$, where $\mathrm{A} = [A_{ik}]$ is independent of the noise variance and is given by (Rattray *et al*, 1998),

$$\mathrm{A}_{ik} = \frac{2}{\pi\sqrt{\Delta}} \left[\mathrm{I} - \frac{1}{\Delta} \left((1+Q_{kk})\mathrm{J}_i\mathrm{J}_i^{\rm T} + (1+Q_{ii})\mathrm{J}_k\mathrm{J}_k^{\rm T} - Q_{ik}(\mathrm{J}_i\mathrm{J}_k^{\rm T} + \mathrm{J}_k\mathrm{J}_i^{\rm T}) \right) \right] , \tag{5.2}$$

with $\Delta = (1+Q_{ii})(1+Q_{kk}) - Q_{ik}^2$.

Recall the general form of the Hessian which was defined in equation (3.1). The Fisher information is also written as the sum of an identity matrix and outer products of weight vectors, but only student weight vectors and student-student overlaps are required here. This is because the average in equation (5.1) does not involve the teacher mapping. The Fisher information matrix should therefore be easier to determine, as only the input distribution is required. Also, although we require our noise model to be correct in order to ensure asymptotic optimality for a sufficiently complex student network, we will see that knowledge of the noise variance is not required.

If the input distribution is Gaussian, then the Fisher information is as defined above. For $K \ll N$ inversion can be achieved efficiently by partitioning (Yang & Amari, 1997) as described in appendix A.1 for the Hessian. Yang & Amari also discuss methods for preprocessing the training examples when the inputs are non-Gaussian, so that the pre-processed inputs approximate a whitened Gaussian process. However, if the input distribution is far from Gaussian then a different approach will be required for inversion. Here we will simply assume that the inputs come from a Gaussian distribution in order to determine the efficacy of natural gradient learning compared with standard gradient descent.

5.2 Dynamics

The weights to hidden node i are updated as follows,

$$\mathbf{J}_i^{\mu+1} = \mathbf{J}_i^{\mu} + \frac{\eta}{N} \sum_{j=1}^{K} \mathbf{A}_{ij}^{-1} \delta_j^{\mu} \boldsymbol{\xi}^{\mu} \ , \tag{5.3}$$

where $\delta_i^{\mu} \equiv g'(x_i^{\mu})[\sum_{n=1}^{M} g(y_n^{\mu}) - \sum_{j=1}^{K} g(x_j^{\mu}) + \rho^{\mu}]$. Notice that the noise variance does not appear explicitly in the above expression, since the noise dependence of the inverse Fisher information matrix and the log-likelihood have cancelled. The derivation of the dynamics closely follows the result for on-line Newton's method and a full discussion will be given elsewhere (Rattray *et al*, 1998; Rattray & Saad, 1998).

Although our equations of motion are sufficient to describe learning for arbitrary system size, the number of order parameters is $\frac{1}{2}K(K-1)+KM$ so that the numerical integration soon becomes rather cumbersome as K and M grow and analysis becomes difficult. To obtain generic results in terms of system size we therefore exploit symmetries which appear in the dynamics for isotropic tasks and structurally matched student and teacher ($K = M$ and $T = T\delta_{nm}$). In this case we define a four dimensional system via $Q_{ij} = Q\delta_{ij} + C(1-\delta_{ij})$ and $R_{in} = R\delta_{in} + S(1-\delta_{in})$ which can be used to study the dynamics for arbitrary K and T. In (Rattray & Saad, 1998) we show how the Fisher information matrix can be inverted for this reduced dimensionality system and the resulting equations of motion are also given there.

As was the case for standard gradient descent (see fig. 1(a)), the dynamics is characterized by two major phases of learning. Initially, the order parameters are trapped in an unstable fixed point characterized by a lack of differentiation between different teacher nodes, the symmetric phase. If the teacher is deterministic then the generalization error eventually converges to zero exponentially, unless the learning rate is chosen too large. If the teacher is corrupted by noise then the learning rate must be annealed in order for the generalization error to decay. As for on-line Newton's method, the fastest decay for natural gradient learning is achieved by setting the learning rate to $1/\alpha$ and the analysis in section 3.4 is equally applicable to natural gradient learning since the methods are asymptotically equivalent.

Unfortunately, even for standard gradient descent an analytical study of the symmetric phase is only possible for small learning rates, which are often far from optimal (Saad & Solla, 1995). Such an approach is not appropriate for realistic learning rates and often gives misleading results since it is the diffusion terms in the dynamics (quadratic in the learning rate) which set the appropriate learning time scale. It is also unclear how to proceed for natural gradient learning even in this limit, since the Fisher information is singular at the fixed point considered by Saad & Solla (1995) and the simplifications used by them are no longer appropriate (Rattray *et al*, 1998). In order to compare

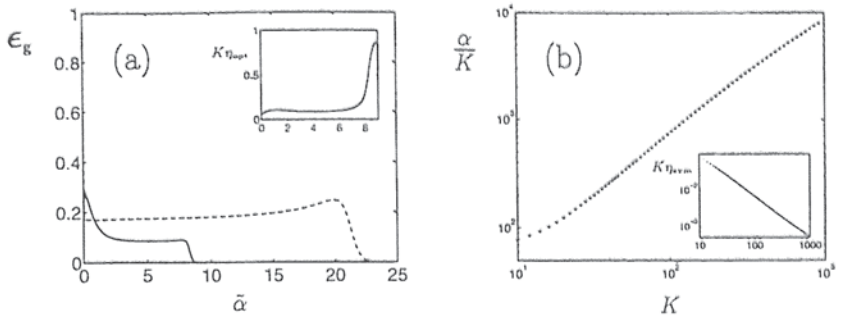

Fig. 4: In (a) the generalization error is shown for optimal natural gradient learning (solid line) and optimal gradient descent (dashed line) for $K = 10$ (we define $\tilde{\alpha} = 10^{-2}\alpha$). The inset shows the optimal learning rate for natural gradient learning. In (b) the time required for optimal natural gradient learning to reach a generalization error of $10^{-4}K$ is shown as a function of K on a log-log scale. The inset shows the optimal learning rate within the symmetric phase. In both (a) and (b) we used $T = 1$, zero noise and initial conditions $R = 10^{-3}$, $Q = U[0, 0.5]$ and $S = C = 0$. Adapted from (Rattray *et al*, 1998).

transient performance with gradient descent for larger learning rates we apply a recent method for determining optimal time-dependent learning rates under the present formalism (Saad & Rattray, 1997, this volume). This allows us to compare the methods at their optimal settings and to determine scaling laws for learning time in terms of task complexity. We note here that the maximal learning rate during the symmetric phase, above which good performance is impossible, is typically close to the optimal value.

Fig. 4 shows results for the optimal learning rate dynamics. In fig. 4(a) we compare the generalization error evolution with the gradient descent result for $K = 10$, showing a significant reduction in learning time. The inset shows the optimal learning rate for natural gradient learning in this case. Notice that we do not consider the effects of noise in this example, since noise will usually be of secondary importance during the symmetric phase, typically resulting in a slightly lengthened and raised plateau. In fig. 4(b) we show the time required to reach a generalization error of $10^{-4}K$ as a function of K. This indicates a scaling law of K^2 for the length of the symmetric plateau (which dominates the learning time) while the corresponding result for gradient descent is $K^{\frac{9}{2}}$ (West & Saad, 1997). The inset shows that the optimal learning rate within the symmetric phase scales as K^{-2} which contrasts with a value of $K^{-\frac{5}{2}}$ for gradient descent. The escape time for the adaptive gradient learning rule studied by West & Saad scales as $K^{\frac{5}{2}}$ which is also worse than for natural

gradient learning.

Our results indicate a significant improvement over gradient descent, which increases with task complexity. However, these results are for the specific case of an isotropic, structurally matched teacher. Numerical studies suggest that improvement can be expected over a larger class of problems but it seems difficult to determine generic results without significantly restricting the class of teacher mappings.

6 Conclusion

We have studied a number of second order on-line algorithms for training multi-layer neural networks, using a statistical mechanics formalism which is appropriate when the input dimension is much larger the the number of hidden units. The first algorithm considered is on-line Newton's method, which is obtained by pre-multiplying the gradient with the inverse Hessian. We find that an unstable fixed point of the gradient descent dynamics becomes stable under this learning rule, which is therefore only useful at late times. Asymptotically, this rule is known to provide optimal performance and we compare the asymptotic decay of the generalization error with the result for gradient descent, for isotropic and structurally matched learning ($K = M$ and $T_{nm} = T\delta_{nm}$). We find that the advantage of using curvature information is most pronounced for small T.

In practice the Hessian can not be determined on-line and inversion is expensive. Matrix momentum provides a practical approximation to on-line Newton's method, since it allows efficient inversion of the Hessian and can be implemented using a very noisy approximation to the Hessian. We first analyse the dynamics of idealized matrix momentum, using the true Hessian, and find that the method converges onto the dynamics of on-line Newton's method in an appropriate limit. Using a single pattern approximation to the Hessian, we find that good asymptotic performance is possible but with some sensitivity to parameter choice, due to noise in the Hessian estimate. More work is required to determine optimal and maximal parameters for this algorithm.

Hessian based methods are inappropriate during the transient phases of learning, since they do not guarantee convergence to a minimum of the generalization error. A principled alternative is to use Amari's natural gradient learning algorithm, which defines a Reimannian metric in the student parameter space. Our analysis of this algorithm points to a significant advantage over gradient descent during the transients of learning, as well as optimal asymptotic performance given a sufficiently rich model. We find improved power law scaling of learning time against task complexity.

The natural gradient learning algorithm requires an inversion of the Fisher information matrix which can be achieved efficiently for Gaussian, or near-

Gaussian, inputs (Yang & Amari, 1997). However, in some cases the inputs will not be close to Gaussian and an efficient method of inversion will be required. Matrix momentum provides a possible inversion method and a natural extension of the present analysis would be to study this case.

Acknowledgements
This work was supported by the EPSRC grant GR/L19232. We would like to thank all the participants of the on-line learning themed week at the Newton Institute for many useful and enlightening discussions.

Appendices

A The Hessian

We define each entry of the Hessian $\mathbf{H} = [H_{i\alpha,k\beta}]$ where $1 \leq i,k \leq K$ and $1 \leq \alpha,\beta \leq N$,

$$H_{i\alpha,k\beta} = \left\langle \frac{\partial^2 \epsilon_{\mathrm{J}}(\xi,\zeta)}{\partial J_{i\alpha}\partial J_{k\beta}} \right\rangle_{\{\xi\}} = \frac{\partial^2 \epsilon_g}{\partial J_{i\alpha}\partial J_{k\beta}} , \tag{A.1}$$

with $\mathbf{J}_i = [J_{i\alpha}]$. To calculate these derivatives we use the chain rule,

$$\frac{\partial \epsilon_g}{\partial J_{i\alpha}} = \sum_{j\geq k} \frac{\partial Q_{jk}}{\partial J_{i\alpha}} \frac{\partial \epsilon_g}{\partial Q_{jk}} + \sum_{jn} \frac{\partial R_{jn}}{\partial J_{i\alpha}} \frac{\partial \epsilon_g}{\partial R_{jn}} , \tag{A.2}$$

where

$$\frac{\partial Q_{jk}}{\partial J_{i\alpha}} = \delta_{ij} J_{k\alpha} + \delta_{ik} J_{j\alpha} , \quad \frac{\partial R_{jn}}{\partial J_{i\alpha}} = \delta_{ij} B_{n\alpha} .$$

Applying the chain rule twice provides us with equation (3.1) in the main text.

A.1 Inversion by partitioning

The Hessian is defined in equation (3.1) and using the block form it is natural to calculate the inverse by partitioning. We will consider the simplest multi-layer student with $K = 2$, although it would be straightforward to iteratively partition for larger K. We write,

$$\mathbf{H} = \begin{pmatrix} \mathrm{D} & \mathrm{E} \\ \mathrm{E}^{\mathrm{T}} & \mathrm{F} \end{pmatrix} , \quad \mathbf{H}^{-1} = \begin{pmatrix} \mathrm{X} & \mathrm{Y} \\ \mathrm{Y}^{\mathrm{T}} & \mathrm{Z} \end{pmatrix} . \tag{A.3}$$

Each block in the inverse can be determined from the following identities,

$$\begin{aligned} \mathrm{X} &= (\mathrm{D} - \mathrm{E}\mathrm{F}^{-1}\mathrm{E}^{\mathrm{T}})^{-1} , \\ \mathrm{Y} &= -(\mathrm{F}^{-1}\mathrm{E}^{\mathrm{T}}\mathrm{X})^{\mathrm{T}} , \\ \mathrm{Z} &= -\mathrm{E}^{-1}\mathrm{D}\mathrm{Y} . \end{aligned} \tag{A.4}$$

Define $\mathrm{S} = (\mathrm{J}_1, \mathrm{J}_2, \mathrm{B}_1, \mathrm{B}_2, \ldots, \mathrm{B}_M)$ where M is the number of hidden nodes in the teacher. Each of the above block matrices takes the same general form; for example let,

$$\mathrm{D} = \mathrm{d}\,(\mathrm{I} + \mathrm{S}\Phi\mathrm{S}^{\mathrm{T}}) \;, \quad \mathrm{F} = \mathrm{f}\,(\mathrm{I} + \mathrm{S}\Psi\mathrm{S}^{\mathrm{T}}) \tag{A.5}$$

where Φ and Ψ are $2+M$ dimensional square matrices while d and f are scalar coefficients. The exact expressions can be found for each block by comparing terms with equation (3.1). The expressions in equation (A.4) can be calculated by repeated application of the following two identities,

$$\mathrm{DF} = \mathrm{d}\,\mathrm{f}\,\left(\mathrm{I} + \mathrm{S}(\Phi + \Psi + \Phi\mathrm{C}\Psi)\mathrm{S}^{\mathrm{T}}\right) \;, \tag{A.6}$$

$$\mathrm{d}\,\mathrm{D}^{-1} = \left(\mathrm{I} - \mathrm{S}(\mathrm{I} + \Phi\mathrm{C})^{-1}\Phi\mathrm{S}^{\mathrm{T}}\right) \;, \tag{A.7}$$

where non-bold I is the $2 + M$ dimensional identity matrix. Here, we have defined C to be the covariance matrix, or matrix of order parameters:

$$\mathrm{C} = \mathrm{S}^{\mathrm{T}}\mathrm{S} = \begin{pmatrix} Q & R \\ R^{\mathrm{T}} & T \end{pmatrix} \;, \tag{A.8}$$

with $Q_{2\times 2} = [Q_{ik}]$, $R_{2\times M} = [R_{in}]$ and $T_{M\times M} = [T_{nm}]$.

The exact expressions for $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ are not presented here as they are rather cumbersome. In fact, we never require explicit expressions in our implementation and we find it simpler to solve equations (A.4) as a sequence of transformations using identities (A.6) and (A.7). For all K the inverse Hessian takes the same general form described by equation (3.2).

A.2 Single pattern approximation

Under the single pattern approximation we no longer average the expression in equation (A.1) over inputs. Each block in the unaveraged Hessian is defined,

$$\frac{\partial^2 \epsilon_{\mathrm{J}}(\xi,\zeta)}{\partial \mathrm{J}_i \partial \mathrm{J}_k} = \xi\,\xi^{\mathrm{T}}\left(\delta_{ik}\, g''(x_i)\Big[\sum_j g(x_j) - \sum_n g(y_n) - \rho\Big] + g'(x_i)g'(x_k)\right) . \tag{A.9}$$

Notice that we could have derived equation (3.1) directly by averaging the above quantity over inputs. However, expanding in terms of the generalization error provides a much simpler expression.

A.3 Asymptotic Hessian

For realizable rules $\mathrm{J} \to \mathrm{B}$ asymptotically and the Hessian is much simplified. Instead of using the definition in equation (3.1) it is simpler to start from the

unaveraged Hessian in equation (A.9), replace every J by a B and average over the inputs and noise. In this case we find for each block,

$$\mathrm{H}_{ik}|_{\mathrm{J}=\mathrm{B}} = \left\langle g'(y_i)g'(y_k)\,\boldsymbol{\xi}\,\boldsymbol{\xi}^{\mathrm{T}}\right\rangle_{\{\xi\}} . \tag{A.10}$$

Recall that $g(x) = \mathrm{erf}(x/\sqrt{2})$ and the inputs come from a Gaussian distribution with zero mean and unit variance, in which case,

$$\mathrm{H}_{ik} = \frac{2}{\pi}\int \frac{\mathrm{d}\boldsymbol{\xi}}{\sqrt{2\pi}}\,\boldsymbol{\xi}\,\boldsymbol{\xi}^{\mathrm{T}} \exp\left[-\tfrac{1}{2}\boldsymbol{\xi}^{\mathrm{T}}(\mathrm{I}+\mathrm{B}_i\mathrm{B}_i^{\mathrm{T}}+\mathrm{B}_k\mathrm{B}_k^{\mathrm{T}})\boldsymbol{\xi}\right] . \tag{A.11}$$

Completing this Gaussian integral and using the fact that $\mathrm{B}_n^{\mathrm{T}}\mathrm{B}_m = T\delta_{nm}$ for an isotropic teacher we find,

$$\begin{aligned}
\mathrm{H}_{ii} &= \frac{2}{\pi\sqrt{1+2T}}\left(\mathrm{I}-\frac{2\mathrm{B}_i\mathrm{B}_i^{\mathrm{T}}}{1+2T}\right) , \\
\mathrm{H}_{i\neq k} &= \frac{2}{\pi(1+T)}\left(\mathrm{I}-\frac{\mathrm{B}_i\mathrm{B}_i^{\mathrm{T}}+\mathrm{B}_k\mathrm{B}_k^{\mathrm{T}}}{1+T}\right) .
\end{aligned} \tag{A.12}$$

A.4 Asymptotic Inversion

For realizable rules we can invert the Hessian asymptotically for any K. The asymptotic form for each block of H can be written,

$$\mathrm{H}_{ik} = (\mathrm{a}\,\delta_{ik}+\mathrm{b})\,\mathrm{I} + (\mathrm{c}\,\delta_{ik}+\mathrm{d})\,\mathrm{B}_i\mathrm{B}_i^{\mathrm{T}} + \mathrm{d}\,\mathrm{B}_k\mathrm{B}_k^{\mathrm{T}} , \tag{A.13}$$

where,

$$\begin{aligned}
\mathrm{a} &= \frac{2}{\pi\sqrt{1+2T}} - \frac{2}{\pi(1+T)} , \quad \mathrm{b} = \frac{2}{\pi(1+T)} , \\
\mathrm{c} &= \frac{4}{\pi(1+T)^2} - \frac{4}{\pi(1+2T)^{\frac{3}{2}}} , \quad \mathrm{d} = -\frac{2}{\pi(1+T)^2} .
\end{aligned}$$

Block (i,k) in the inverse of H is then given by,

$$\mathrm{H}_{ik}^{-1} = \left(\frac{1}{\mathrm{a}}\,\delta_{ik} - \frac{\mathrm{b}}{\mathrm{a}(\mathrm{a}+\mathrm{b}K)}\right)\mathrm{I} + \sum_{n=1}^{K}\Gamma_{ik}^{n}\mathrm{B}_n\mathrm{B}_n^{\mathrm{T}} , \tag{A.14}$$

Substituting these expressions into the definition of an inverse and using the orthogonality of the teacher weight vectors we obtain a matrix equation for the K dimensional square matrix $\Gamma^n = [\Gamma_{ik}^n]$,

$$\Gamma^n = \mathrm{P}^{-1}\mathrm{X} \tag{A.15}$$

where,

$$\begin{aligned}
\mathrm{P} &= \mathrm{a}\,\mathrm{I} + \begin{pmatrix} \mathrm{e}_n \\ \mathrm{u} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} cT & -\mathrm{d}T \\ -\mathrm{d}T & \mathrm{b} \end{pmatrix} \begin{pmatrix} \mathrm{e}_n \\ \mathrm{u} \end{pmatrix} , \\
\mathrm{X} &= \frac{1}{\mathrm{a}} \begin{pmatrix} \mathrm{e}_n \\ \mathrm{u} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} -\mathrm{c} & (\mathrm{da}-\mathrm{bc})/(\mathrm{a}+\mathrm{b}K) \\ \mathrm{d} & -\mathrm{db}/(\mathrm{a}+\mathrm{b}K) \end{pmatrix} \begin{pmatrix} \mathrm{e}_n \\ \mathrm{u} \end{pmatrix} .
\end{aligned}$$

Here, we have defined $\mathbf{e}_n$ to be a K dimensional row vector with a one in the nth element and zeros everywhere else, $\mathbf{u}$ is a row vector of ones and I is the K dimensional identity. Solving for Γ^n we find,

$$\Gamma^n = \frac{1}{\Delta} \begin{pmatrix} \mathbf{e}_n \\ \mathbf{u} \end{pmatrix}^{\mathrm{T}} \Theta \begin{pmatrix} \mathbf{e}_n \\ \mathbf{u} \end{pmatrix} , \tag{A.16}$$

where,

$$\begin{aligned} \Theta &= \begin{pmatrix} K(\mathrm{d}^2 T - \mathrm{bc}) - \mathrm{ac} & \mathrm{d}^2 T - \mathrm{bc} - \mathrm{ad} \\ \mathrm{d}^2 T - \mathrm{bc} - \mathrm{ad} & (\mathrm{d}^2 T(\mathrm{a}+\mathrm{b}) - 2\mathrm{abd} - \mathrm{b}^2\mathrm{c})/(\mathrm{a}+\mathrm{b}K) \end{pmatrix} , \\ \Delta &= \mathrm{a}^2(\mathrm{a}+\mathrm{b}K) + \mathrm{a}^2 T(\mathrm{c}-2\mathrm{d}) + \mathrm{a}T(K-1)(\mathrm{cb}-\mathrm{d}^2) . \end{aligned}$$

B Derivatives of the generalization error

Saad & Solla (1995) calculate the generalization error,

$$\epsilon_g = \frac{1}{\pi} \left[\sum_{ik} \arcsin\left(\frac{Q_{ik}}{\sqrt{1+Q_{ii}}\sqrt{1+Q_{kk}}} \right) + \sum_{nm} \arcsin\left(\frac{T_{nm}}{\sqrt{1+T_{nn}}\sqrt{1+T_{mm}}} \right) - 2 \sum_{in} \arcsin\left(\frac{R_{in}}{\sqrt{1+Q_{ii}}\sqrt{1+T_n n}} \right) \right] . \tag{B.1}$$

We require the following derivatives,

$$\frac{\partial \epsilon_g}{\partial Q_{ik}} = \frac{2-\delta_{ik}}{\pi\sqrt{(1+Q_{ii})(1+Q_{kk})-Q_{ik}^2}} + \frac{\delta_{ik}}{\pi(1+Q_{ii})} \sum_n \frac{R_{in}}{\sqrt{(1+T_{nn})(1+Q_{ii})-R_{in}^2}} - \frac{\delta_{ik}}{\pi(1+Q_{ii})} \sum_j \frac{Q_{ij}}{\sqrt{(1+Q_{ii})(1+Q_{jj})-Q_{ij}^2}} , \tag{B.2}$$

$$\frac{\partial^2 \epsilon_g}{\partial Q_{jl} \partial R_{km}} = \frac{\delta_{jl}\delta_{jk}(1+T_{mm})}{\pi((1+T_{mm})(1+Q_{jj})-R_{jm}^2)^{\frac{3}{2}}} , \tag{B.3}$$

$$\frac{\partial^2 \epsilon_g}{\partial R_{jm} \partial R_{in}} = -\frac{2\delta_{ij}\delta_{nm}R_{jm}}{\pi((1+T_{mm})(1+Q_{jj})-R_{jm}^2)^{\frac{3}{2}}} , \tag{B.4}$$

$$\begin{aligned} \frac{\partial^2 \epsilon_g}{\partial Q_{jl} \partial Q_{kr}} &= \frac{(\delta_{jl}-2)((1+Q_{ll})\delta_{kj}\delta_{rj} + (1+Q_{jj})\delta_{kl}\delta_{rl} - 2Q_{jl}(\delta_{jk}\delta_{lr} + \delta_{jr}\delta_{lk}(1-\delta_{jl})))}{2\pi((1+Q_{jj})(1+Q_{ll})-Q_{jl}^2)^{\frac{3}{2}}} \\ &+ \frac{\delta_{jl}\delta_{jk}\delta_{jr}}{2\pi(1+Q_{jj})^2} \sum_i Q_{ij} \frac{3(1+Q_{ii})(1+Q_{jj})-2Q_{ij}^2}{((1+Q_{ii})(1+Q_{jj})-Q_{ij}^2)^{\frac{3}{2}}} \\ &- \frac{\delta_{jl}\delta_{jk}\delta_{jr}}{2\pi(1+Q_{jj})^2} \sum_n R_{jn} \frac{3(1+Q_{jj})(1+T_{nn})-2R_{jn}^2}{((1+T_{nn})(1+Q_{jj})-R_{jn}^2)^{\frac{3}{2}}} \\ &- \frac{\delta_{jl}}{2\pi} \left[\frac{2\delta_{jr}(1+Q_{kk})-\delta_{rk}Q_{kj}}{((1+Q_{kk})(1+Q_{jj})-Q_{kj}^2)^{\frac{3}{2}}} + \frac{2\delta_{jk}(1-\delta_{jr})(1+Q_{rr})}{((1+Q_{rr})(1+Q_{jj})-Q_{rj}^2)^{\frac{3}{2}}} \right] . \end{aligned} \tag{B.5}$$

C Equations of motion

In this appendix we provide the large N equations of motion for each algorithm. In each case we define a new time variable $\alpha = \mu/N$ to be the normalized number of patterns presented so far.

C.1 On-line Newton's Method

$$
\begin{aligned}
\frac{\mathrm{d}R_{in}}{\mathrm{d}\alpha} &= \eta \sum_{k=1}^{K} \alpha_{ik} \left(\phi_{kn} + \sum_{j=1}^{K} R_{jn} \left[\sum_{l=1}^{K} \Theta_{jl}^{ik} \psi_{kl} + \sum_{m=K+1}^{M+K} \Theta_{jm}^{ik} \phi_{km} \right] \right. \\
&\left. + \sum_{p=K+1}^{M+K} T_{pn} \left[\sum_{l=1}^{K} \Theta_{pl}^{ik} \psi_{kl} + \sum_{m=K+1}^{M+K} \Theta_{pm}^{ik} \phi_{km} \right] \right) , \\
\frac{\mathrm{d}Q_{ir}}{\mathrm{d}\alpha} &= \eta \sum_{k=1}^{K} \alpha_{ik} \left(\psi_{kr} + \sum_{j=1}^{K} Q_{jr} \left[\sum_{l=1}^{K} \Theta_{jl}^{ik} \psi_{kl} + \sum_{m=K+1}^{M+K} \Theta_{jm}^{ik} \phi_{km} \right] \right. \\
&\left. + \sum_{n=K+1}^{M+K} R_{rn} \left[\sum_{l=1}^{K} \Theta_{nl}^{ik} \psi_{kl} + \sum_{m=K+1}^{M+K} \Theta_{nm}^{ik} \phi_{km} \right] \right) + \\
&\eta \sum_{k=1}^{K} \alpha_{rk} \left(\psi_{ki} + \sum_{j=1}^{K} Q_{ji} \left[\sum_{l=1}^{K} \Theta_{jl}^{rk} \psi_{kl} + \sum_{m=K+1}^{M+K} \Theta_{jm}^{rk} \phi_{km} \right] \right. \\
&\left. + \sum_{n=K+1}^{M+K} R_{in} \left[\sum_{l=1}^{K} \Theta_{nl}^{rk} \psi_{kl} + \sum_{m=K+1}^{M+K} \Theta_{nm}^{rk} \phi_{km} \right] \right) + \eta^2 \sum_{k=1}^{K} \sum_{l=1}^{K} \alpha_{ik} \alpha_{rl} \upsilon_{kl} \ .
\end{aligned}
\tag{C.1}
$$

Here, Θ is defined by equation (3.2) and we have defined $\phi_{in} \equiv \langle \delta_i y_n \rangle_{\{\xi\}}$, $\psi_{ik} \equiv \langle \delta_i x_k \rangle_{\{\xi\}}$ and $\upsilon_{ik} \equiv \langle \delta_i \delta_k \rangle_{\{\xi\}}$. The explicit expressions for ϕ_{in}, ψ_{ik}, υ_{ik} depend exclusively on the overlaps Q, R and T and are given in (Saad & Solla, 1995; Saad & Rattray, this volume).

C.2 Idealized Matrix Momentum

For matrix momentum using the true Hessian we find,

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

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 order parameters as covariances and all averages can be calculated in closed form, as described in (Saad & Solla, 1995; Saad & Rattray, this volume). The second derivatives of the generalization error are given in appendix B.

C.3 Single Pattern Matrix Momentum

For matrix momentum using a single pattern approximation to the Hessian we find,

$$\begin{aligned} \frac{dQ_{ik}}{d\alpha} &= E_{ik} + E_{ki} , \qquad \frac{dR_{in}}{d\alpha} = D_{in} , \\ \frac{dC_{ik}}{d\alpha} &= k\langle(\eta_\alpha\delta_i - \phi_i)z_k + (\eta_\alpha\delta_k - \phi_k)z_i\rangle + k^2\langle(\eta_\alpha\delta_i - \phi_i)(\eta_\alpha\delta_k - \phi_k)\rangle , \\ \frac{dD_{in}}{d\alpha} &= k\langle(\eta_\alpha\delta_i - \phi_i)y_n\rangle , \qquad \frac{dE_{ik}}{d\alpha} = C_{ik} + k\langle(\eta_\alpha\delta_k - \phi_k)x_i\rangle . \end{aligned} \tag{C.3}$$

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 carried out explicitly to provide a closed set of equations of motion. The following identities are required (recall that $g(x) = \mathrm{erf}\,(x/\sqrt{2})$),

$$\begin{aligned} \int \frac{\mathrm{dx}}{\sqrt{|\mathrm{C}|(2\pi)^n}}\, g(x_i)\, \mathrm{e}^{-\frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{x}+\mathbf{d}^{\mathrm{T}}\mathbf{x}} &= g\left(\frac{\sum_k C_{ik}d_k}{\sqrt{1+C_{ii}}}\right)\mathrm{e}^{\frac{1}{2}\mathbf{d}^{\mathrm{T}}\mathbf{C}\mathbf{d}} , \qquad \text{(C.4)} \\ \int \frac{\mathrm{dx}}{\sqrt{|\mathrm{C}|(2\pi)^n}}\, g(x_i)\, g(x_k)\, \mathrm{e}^{-\frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{x}} &= \frac{2}{\pi}\arcsin\left(\frac{C_{ik}}{\sqrt{1+C_{ii}}\sqrt{1+C_{kk}}}\right) , \end{aligned}$$

where $\mathbf{C}$ is an $n \times n$ symmetric matrix. Factors involving the components of $\mathbf{x}$ can be brought down from the exponent of the integrand by differentiation.

References

Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*, 10, 251–276.

Barber, D., Saad, D., Sollich, P. (1996). Finite-size effects in online learning of multilayer neural networks. *Europhysics Letters*, 34, 151–156.

Biehl, M., Schwarze, H. (1995). Learning by online gradient descent. *Journal of Physics A*, 28, 643–656.

Biehl, M., Riegler, P., Wöhler, C. (1996). Transient dynamics of online learning in two layered neural networks. *J. Phys. A*, 29, 4769–4780.

Bishop, C. M. (1995) *Neural networks for pattern recognition.* Oxford University Press, Oxford, UK.

LeCun, Y., Simard, P. Y., Pearlmutter, B. (1993). Approximation by superpositions of a sigmoid function.*Advances in Neural Information Processing Systems 5*, ed. C.L. Giles, S.J. Hanson and J.D. Cowan (San Mateo, CA: Morgan Kaufmann) .

Orr, G. B. (1995). *Dynamics and Algorithms for Stochastic Search.* PhD. Dissertation, Oregon Graduate Institute of Science and Technology.

Leen, T.K., Orr, G.B. (1994). Optimal stochastic search and adaptive momentum. *Advances in Neural Information Processing Systems 6*, ed J. D. Cowan, G. Tesauro and J. Alspector (San Francisco, CA: Morgan Kaufmann)

Orr, G. B., Leen, T. K. (1997). Using curvature information for fast stochastic search. *Advances in Neural Information Processing Systems 9*, ed M. C. Mozer, M. I. Jordan and T. Petsche (Cambridge, MA: MIT Press)

Rattray, M., Saad, D. (1998). An analysis of on-line learning training with optimal learning rates. *Phys. Rev. E* in press.

Rattray, M., Saad, D., Amari, S. (1998). Natural gradient descent for on-line learning, *unpublished.*

Riegler, P., Biehl, M. (1995). Online backpropagation in two layered neural networks. *J. Phys. A*, 28, L507–L513.

Roy, S., Shynk, J. J. (1990). Analysis of the momentum LMS algorithm. *IEEE transactions on acoustics, speech and signal processing*, 38, 2088–2098.

Saad, D., Rattray, M. (1997). Globally optimal parameters for on-line learning in multilayer neural networks. *Phys. Rev. Lett.*, 79, 2578–2581.

Saad, D., Rattray, M. (1998). Optimal on-line learning in multilayer neural networks *in this volume.*

Saad, D., Solla, S. A. (1995). Exact solution for online learning in multilayer neural networks. *Phys. Rev.Lett.*, 74, 4337–4340 , Online learning in soft committe machines. *Phys. Rev. E*, 52, 4225–4243.

West, A. H. L., Saad, D. (1997). On-line learning with adaptive back-propagation in two-layer networks. *Phys. Rev. E*, 56, 3426–3445.

Wiegerinck, W., Komoda, A., Heskes, T. (1994). Stochastic dynamics of learning with momentum in neural networks. *J. Phys. A*, 27, 4425–4437.

Yang, H. Y., Amari, S. (1998). The efficiency and the robustness of natural gradient descent learning rule. *Advances in Neural Information Processing Systems 10*, ed M. I. Jordan, M. J. Kearns and S. A. Solla (Cambridge, MA: MIT Press).