Accelerated Gradient Descent by Factor-Centering Decomposition

Technical Report IDSIA-33-98

Nicol N. Schraudolph
`nic@idsia.ch`

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

May 20, 1998

Abstract

Gradient factor centering is a new methodology for decomposing neural networks into *biased* and *centered* subnets which are then trained in parallel. The decomposition can be applied to any pattern-dependent factor in the network's gradient, and is designed such that the subnets are more amenable to optimization by gradient descent than the original network: biased subnets because of their simplified architecture, centered subnets due to a modified gradient that improves conditioning. The architectural and algorithmic modifications mandated by this approach include both familiar and novel elements, often in prescribed combinations. The framework suggests for instance that *shortcut connections* — a well-known architectural feature — should work best in conjunction with *slope centering,* a new technique described herein. Our benchmark experiments bear out this prediction, and show that factor-centering decomposition can speed up learning significantly without adversely affecting the trained network's generalization ability.

1 Introduction

Due to the chain and product rules for differentiation, the gradient of an objective function E with respect to any given adjustable parameter w of a neural network — that is, a differentiable (semi-)parametric model — will typically be composed of a number of factors. Depending on the network's architecture, these factors may include input or hidden node activities, error signals (gradients *wrt.* nodes), slopes (derivatives) of transfer functions, gating factors, and so forth.

For any given factor a_k, the gradient can be split into two terms — one that contains the average value $\langle a_k \rangle$ over training patterns, the other featuring the fluctuations of a_k around this average:

$$\begin{aligned} \partial E / \partial w &= a_1 \cdot a_2 \cdot \ldots \cdot a_k \cdot \ldots \\ &= a_1 \cdot a_2 \cdot \ldots \cdot \langle a_k \rangle \cdot \ldots \quad + \quad a_1 \cdot a_2 \cdot \ldots \cdot (a_k - \langle a_k \rangle) \cdot \ldots \end{aligned} \tag{1}$$

The object of gradient factor-centering decomposition is to construct two networks, called the *biased* and *centered* subnets, whose respective gradients correspond to the first and second term in (1). Since the sum of their gradients is equivalent to that of the original network, the two subnets form an equivalent (though re-parametrized) adaptive system when their outputs are added together. Note also that because their gradients have zero covariance with respect to a_k (since $\big\langle \langle a_k \rangle \, (a_k - \langle a_k \rangle) \big\rangle = 0$) there should be minimal interference between them when they are trained in parallel.[1]

The point of this decomposition is that each of the subnets can learn its task faster than the original network. With $\langle a_k \rangle$ remaining constant[2], the biased subnet has a simplified gradient — it learns an inherently easier subtask. For the centered subnet, meanwhile, the problem is likely to have become better-conditioned: to the extent to which $\langle a_k \rangle$ differs from zero, its decimation (by centering a_k) reduces the leading eigenvalue of the Hessian. This permits the centered subnet to learn at a higher rate, and hence converge sooner. Both subnets together, trained in parallel at different rates, can thus be expected to learn faster than the original network.

We typically implement the biased subnet by performing gradient descent on a simplified architecture, and the centered subnet by suitably modifying the gradient descent procedure for the original architecture. It should be noted that this rarely results in two subnets whose respective gradients are *exactly* equal to the two terms in (1). We usually find it more practical — and empirically quite adequate — for the subnets to loosely conform to the roles implied for them by the factor-centering decomposition. The remainder of this paper provides a number of examples.

2 Activity and Error Centering

Consider a linear node $y = \vec{w}^T \vec{x}$, with weight vector $\vec{w}$ adapted by stochastic gradient descent so as to reduce the squared error objective $E = \frac{1}{2}(y - y^*)^2$:

$$\vec{w} \;\leftarrow\; \vec{w} - \eta \cdot \partial E / \partial \vec{w} \;=\; \vec{w} + \eta\, \delta\, \vec{x}\,, \quad \text{where} \quad \delta \;=\; -\,\partial E / \partial y \;=\; y^* - y \tag{2}$$

[1] Some interference may be introduced via correlations of a_k with other gradient factors.

[2] We assume that on the time scale of pattern-dependent activity, weight parameters and averages such as $\langle a_k \rangle$ can be treated as constant. For online algorithms — where averages are typically implemented as exponential traces — this only holds approximately.

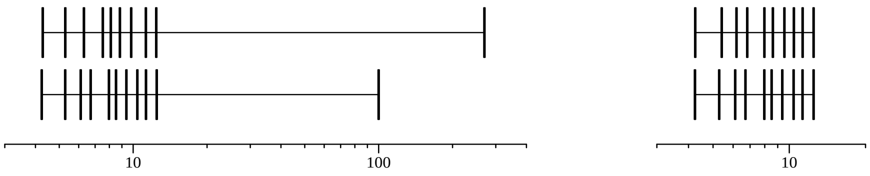

Figure 1: Eigenvalue spectra for the covariance of a 100x10 matrix with uniformly random entries in the range $[0, 1)$. Clockwise from upper left: uncentered, centered *a priori* by subtracting 0.5, centered exactly, and centered with a bias column added.

is the error signal. The convergence time of this system is proportional to the condition number (the ratio between largest and smallest eigenvalue) of the Hessian (Widrow et al., 1976), which here equals the covariance of the inputs: $H = \langle \vec{x}\,\vec{x}^T \rangle$. LeCun et al. (1991) have observed that centering the input vector $\vec{x}$ typically decimates the largest eigenvalue of H, permitting an increase of the learning rate η, and a correspondingly shorter convergence time. Figure 1 illustrates that centering can indeed decrease the condition number (distance between leftmost and rightmost spectral line) of a random covariance matrix dramatically.

From the perspective of factor-centering decomposition, centering $\vec{x}$ clearly sets up a centered subnet — but where is the biased subnet? Our decomposition (1) would suggest an additional weight vector $\vec{v}$ such that

$$y \;=\; \vec{w}^T(\vec{x} - \langle \vec{x} \rangle) \;+\; \vec{v}^T \langle \vec{x} \rangle \;, \tag{3}$$

but since all inputs to $\vec{v}$ are constant, it can be replaced by a single bias weight with a constant input of one. Note that the centered subnet has no bias weights, since bias inputs are rendered ineffective (constant zero) by centering. Our approach thus decomposes the network into bias and non-bias weights, the point being that these two should be trained at different rates (*cf.* Schraudolph & Sejnowski, 1996). Figure 1 (bottom left) illustrates what can happen if this advice is ignored: the bias input can reintroduce the large eigenvalue that centering has laboriously eliminated.

We have suggested before (Schraudolph & Sejnowski, 1996) that one can also center the node's error signal δ. In expectation this is equivalent to centering the inputs:

$$\partial \langle E \rangle / \partial \vec{w} \Big\lfloor_{\delta} \equiv \langle (\delta - \langle \delta \rangle)\, \vec{x} \rangle \;=\; \langle \delta\, \vec{x} \rangle - \langle \delta \rangle \langle \vec{x} \rangle \;=\; \langle \delta\, (\vec{x} - \langle \vec{x} \rangle) \rangle \;\equiv\; \partial \langle E \rangle / \partial \vec{w} \Big\lfloor_{\vec{x}} \tag{4}$$

where $\cdot\lfloor_u$ denotes "centered *wrt.* u". Due to this equivalence, the biased subnet is also the same, *i.e.*, the bias weight. Error centering is a useful addition whenever $\vec{x}$ is centered only approximately, as is for instance the case in online learning.

Although we have introduced them above for the simple case of a linear node, the activity- and error-centering decompositions naturally extend to more complex architectures such as multi-layer perceptrons. Hidden node activity in such networks can be approximately centered *a priori* by the simple expedient of using an antisymmetric activation function such as the hyperbolic tangent. Section 5 gives some empirical results in such a setting.

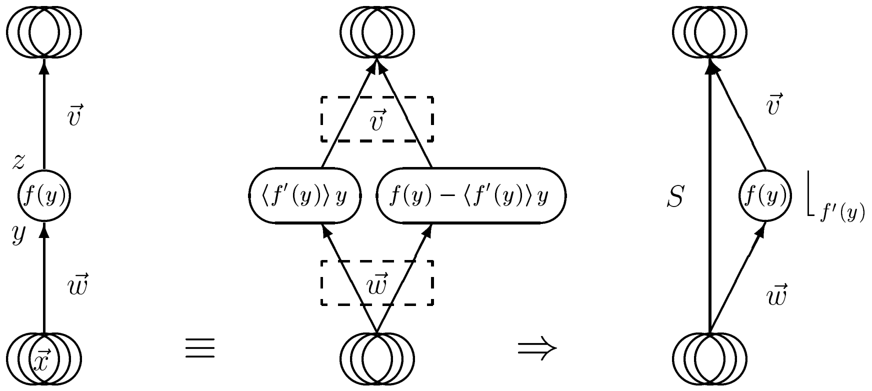

Figure 2: Slope-centering decomposition splits the hidden nodes of a multi-layer perceptron (left) in two, creating an equivalent network (center) which can be simplified to a network with shortcuts S and slope centering in the bypassed nodes (right).

3 Slope Centering

Where activity is passed through a nonlinearity, such as the activation function f of a hidden node in a multi-layer perceptron, an additional factor enters the gradient: the *slope* (derivative) f' of the nonlinearity. Splitting $f(y)$ into the components $\langle f'(y)\rangle\, y$ (biased subnet) and $f(y) - \langle f'(y)\rangle\, y$ (centered subnet) decomposes the gradient in accordance with (1). Noting that the biased subnet is in fact linear (since $\langle f'(y)\rangle \approx$ const.), we can replace it with a matrix S of *shortcut* weights, as illustrated in Figure 2.

We also let the centered subnet use the original activation function $f(y)$ while continuing to center the slope $f'(y)$ in its gradient. This means that the centered subnet is no longer performing gradient descent, and may in fact take steps that systematically *increase* E. These deviations from the gradient, however, fall into the linear subspace that the biased subnet operates in. Since the shortcuts solve a far simpler (linear) subproblem, calling upon them to correct errors introduced by the centered subnet in this fashion does not unduly affect overall performance.

Shortcuts are a popular architectural feature of multi-layer perceptrons, in particular those with more than one hidden layer. They are generally thought to be beneficial to the learning process by providing a linear subnet that a) backpropagates error gradients to preceding layers without the blurring and attenuation associated with the passage through a layer of hidden nodes, and b) frees the bypassed hidden nodes from responsibility for the linear component (now implemented by the shortcuts) of the mapping they are to learn.

We note that while shortcuts render the linear component of the error irrelevant to a hidden node *in principle*, it is slope centering that actually removes this component from its error signal — see (Schraudolph, 1998b) for a more detailed discussion. In short, our work suggests that shortcuts should always be used in conjunction with slope centering, and this is borne out in our empirical findings (see Section 5).

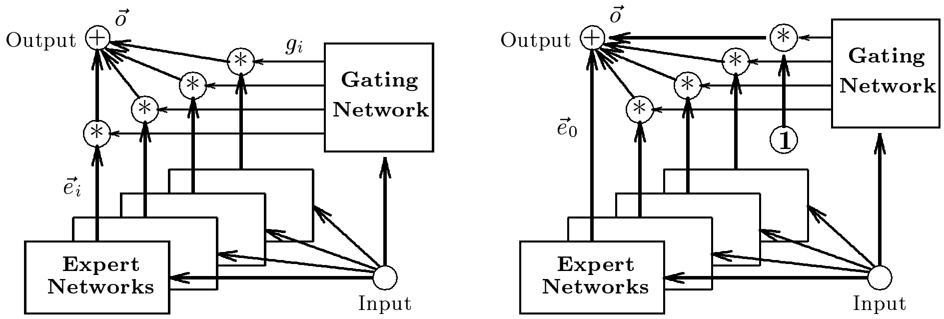

Figure 3: A conventional mixture-of-experts architecture (left), and the factor-centered version with gated bias weights and a uniform background expert (right).

4 Gate Centering

Consider the conventional *mixture-of-experts* architecture shown in Figure 3 (left). The output $\vec{o}$ is a weighted sum of the individual expert networks' outputs $\vec{e}_i$, with the weights g_i (usually constrained to sum to one) computed by a gating network. The error gradient for an expert network contains its corresponding gating factor, and *vice versa*:

$$\partial E/\partial \vec{e}_i \;=\; g_i \cdot \partial E/\partial \vec{o} \quad \text{and} \quad \partial E/\partial g_i \;=\; \vec{e}_i^{\,T} \cdot \partial E/\partial \vec{o} \tag{5}$$

Applying our decomposition to g_i and $\vec{e}_i$, we obtain the gradients

$$\partial E/\partial \vec{e}_i \Big\lfloor_{g_i} \;=\; (g_i - \langle g_i \rangle)\, \partial E/\partial \vec{o} \quad \text{and} \quad \partial E/\partial g_i \Big\lfloor_{\vec{e}_i} \;=\; (\vec{e}_i - \langle \vec{e}_i \rangle)^T \partial E/\partial \vec{o} \tag{6}$$

for the centered subnet, and two biased subnets: a *background expert* whose output $\vec{e}_0$ is always added to $\vec{o}$ (corresponding to a constant gating value), and a vector of *gated bias weights* (corresponding to an expert with constant output). Figure 3 (right) shows the proposed architecture, which we are currently implementing for benchmarking purposes.

5 Empirical Results

We now present empirical results for two benchmarks: the toy problem of symmetry detection in binary patterns, and a difficult vowel recognition task. All experiments used a combination of two acceleration techniques: *vario-η* (Neuneier & Zimmermann, 1998; Zimmermann, 1994, page 48) to normalize local step sizes, and *bold driver* (Lapedes & Farber, 1986; Battiti, 1992) to adjust the global learning rate. Thus any performance advantage for our approach reported thereafter has been realized *on top of* a state-of-the-art accelerated gradient method as control. See (Schraudolph, 1998a) for further details.

5.1 Symmetry Detection Problem

A fully connected 8-8-1 feedforward network was to indicate whether a given binary input was symmetric about its middle axis or not. The network was trained on all

error signals: / activities:	conventional: **mean ± st.d.** / quartiles	direct comparison: # of faster runs			centered: **mean ± st.d.** / quartiles
off-center (0/1)	**669 ± 308** 453/580/852		0 − 100		**97.5 ± 21.8** 82/95.5/109
		0	0 × 35	7	
		100	63 100	93	
centered (-1/1)	**93.1 ± 46.7** 67.5/79.5/94		14 − 84		**65.4 ± 15.9** 57/62/70

Table 1: The number of epochs required to converge to criterion on the 8-bit symmetry detection task, with *vs.* without centering of activity and/or error signals. Runs with identical random seeds are also compared directly (may sum to less than 100 due to ties).

256 patterns using a cross-entropy loss function until the root-mean-square error of its output fell below 0.01. Data was collected over 100 runs, so the standard error of the reported mean time to convergence is $1/\sqrt{100} = 1/10$ its standard deviation.

Table 1 shows that centering either activity or error signals produced an approximate 7-fold increase in convergence speed; the similar magnitude of effect is indicative of the close relationship implied by (4). Note, however, that error centering also cut the coefficient of variation (c.v.) in half, while activity centering left it unchanged. We speculate that this may be the beneficial effect of centering on the *backpropagated* error, which does not occur for activity centering. A further speedup of 50% occurred when both activity and error signals were centered; this can be attributed to the fact that hidden node activity was centered only approximately (*a priori*) by use of the hyperbolic tangent function.

Table 2 shows that adding shortcuts alone was in fact detrimental — it slowed down convergence in over 80% of the runs, and significantly increased the c.v. Subsequent addition of slope centering, however, brought about an almost 3-fold increase in learning speed, and restored the original c.v. of about 1/4. When used together, slope centering and shortcuts never increased convergence time, and on average cut it in half. By contrast, slope centering *without* shortcuts failed to converge about 1/3 of the time. This is no surprise: since slope centering projects the backpropagated gradient into the null space of shortcut weights, the hidden nodes can no longer reduce the linear component of the error signal on their own.

slopes: / topology:	conventional: **mean ± st.d.** / quartiles	direct comparison: # of faster runs			centered: **mean ± st.d.** / quartiles
shortcuts? no	**65.4 ± 15.9** 57/62/70		52 − 48		* **51.6 ± 16.2** 43/64.5/∞
		81	0 × 61	4	
		17	39 99	95	
shortcuts? yes	**90.4 ± 31.1** 69.5/80/102		0 − 100		**33.1 ± 8.6** 28/31/35

* Mean and standard deviation exclude 34 runs which did not converge.

Table 2: The effect of centering slopes and/or adding shortcuts on the symmetry detection task with centered activity and error signals. Results shown as in Table 1.

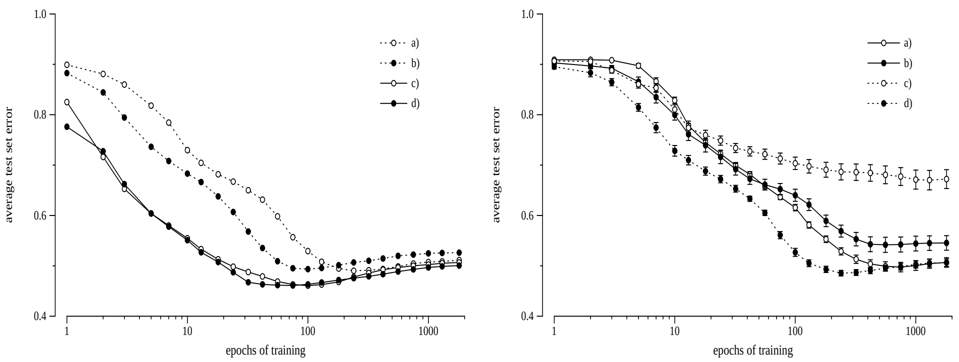

Figure 4: Evolution of the average test set error while learning the vowel recognition task. Left: using activity (solid lines) and/or error (filled marks) centering; right: using slope centering (dashed lines) and/or shortcut weights (filled marks).

5.2 Vowel Recognition Problem

We also tested our approach on a speaker-independent vowel recognition problem (Deterding, 1989) which has been adopted (Robinson, 1989) as a popular neural network benchmark (*e.g.,* Hochreiter & Schmidhuber, 1997; Flake, 1998). The task is to recognize the eleven steady-state vowels of British English, given 10 spectral features of the speech signal. The data consists of 990 patterns: 6 instances for each of the 11 vowels spoken by each of 15 speakers. We followed the conventional split into training (first 8 speakers) and test set (remaining 7), and trained fully connected feedforward networks with 10 inputs, 22 hidden nodes, and 11 outputs by minimization of cross-entropy. After each epoch, generalization ability was measured in terms of the maximum likelihood misclassification rate on the test set — again, see (Schraudolph, 1998a) for further details.

The evolution of the average test set error (over 100 runs) while training with *vs.* without activity and/or error centering is shown in Figure 4 (left).[3] Activity centering had a large beneficial effect both in terms of convergence speed (note the logarithmic scale) and minimum test set error. Error centering, by contrast, did not significantly affect generalization ability, though it did speed up learning. As we would expect from (4), that effect was greater in the absence of activity centering.

Figure 4 (right) shows the effect (averaged over 25 runs) of adding shortcuts and/or slope centering. While shortcut weights alone again worsened generalization performance, in conjunction with slope centering they resulted in faster convergence (by a factor of five), to lower test test errors. As to be expected, the use of slope centering *without* shortcuts again proved ill-advised.

Note that our results on both benchmarks contradict the popular notion that shortcut weights *per se* are beneficial. Gradient factor-centering decomposition suggests that they should be combined with slope centering, and indeed we find empirically that this appears to be the key for making shortcuts genuinely useful.

[3] Error bars were too small here to warrant their display in the graph.

Acknowledgment

This work was supported by the Swiss National Science Foundation under grant numbers 2100–045700.95/1 and 2000–052678.97/1.

References

Battiti, R. (1992). First- and second-order methods for learning: Between steepest descent and Newton's method. *Neural Computation*, *4*(2), 141–166.

Deterding, D. H. (1989). *Speaker Normalisation for Automatic Speech Recognition*. Ph.D. thesis, University of Cambridge.

Flake, G. W. (1998). Square unit augmented, radially extended, multilayer perceptrons. In *Neural Networks: Tricks of the Trade* (Orr & Müller, 1998), pp. 145–163.

Hochreiter, S., & Schmidhuber, J. (1997). Unsupervised coding with LOCOCODE. In *Proceedings of the 7th International Conference on Artificial Neural Networks*, pp. 655–660 Lausanne, Switzerland. Springer Verlag, Berlin.

Lapedes, A., & Farber, R. (1986). A self-optimizing, nonsymmetrical neural net for content addressable memory and pattern recognition. *Physica*, *D 22*, 247–259.

LeCun, Y., Kanter, I., & Solla, S. A. (1991). Eigenvalues of covariance matrices: Application to neural-network learning. *Physical Review Letters*, *66*(18), 2396–2399.

Neuneier, R., & Zimmermann, H. G. (1998). How to train neural networks. In *Neural Networks: Tricks of the Trade* (Orr & Müller, 1998), pp. 373–423.

Orr, G. B., & Müller, K.-R. (Eds.). (1998). *Neural Networks: Tricks of the Trade*, Vol. 1524 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin.

Robinson, A. J. (1989). *Dynamic Error Propagation Networks*. Ph.D. thesis, University of Cambridge.

Schraudolph, N. N. (1998a). Centering neural network gradient factors. In *Neural Networks: Tricks of the Trade* (Orr & Müller, 1998), pp. 207–226. ◁ftp://ftp.idsia.ch/pub/nic/center.ps.gz▷.

Schraudolph, N. N. (1998b). Slope centering: Making shortcut weights effective. In Niklasson, L., Bodén, M., & Ziemke, T. (Eds.), *Proceedings of the 8th International Conference on Artificial Neural Networks*, Perspectives in Neural Computing, pp. 523–528 Skövde, Sweden. Springer Verlag, Berlin. ◁ftp://ftp.idsia.ch/pub/nic/slope.ps.gz▷.

Schraudolph, N. N., & Sejnowski, T. J. (1996). Tempering backpropagation networks: Not all weights are created equal. In Touretzky, D. S., Mozer, M. C., & Hasselmo, M. E. (Eds.), *Advances in Neural Information Processing Systems*, Vol. 8, pp. 563–569. The MIT Press, Cambridge, MA. ◁ftp://ftp.idsia.ch/pub/nic/nips95.ps.gz▷.

Widrow, B., McCool, J. M., Larimore, M. G., & Johnson, Jr., C. R. (1976). Stationary and nonstationary learning characteristics of the LMS adaptive filter. *Proceedings of the IEEE*, *64*(8), 1151–1162.

Zimmermann, H. G. (1994). Neuronale Netze als Entscheidungskalkül. In Rehkugler, H., & Zimmermann, H. G. (Eds.), *Neuronale Netze in der Ökonomie: Grundlagen und finanzwirtschaftliche Anwendungen*, pp. 1–87. Vahlen Verlag, Munich.