Slope Centering: Making Shortcut Weights Effective*

May 9, 1998

Nicol N. Schraudolph
nic@idsia.ch

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

Abstract

Shortcut connections are a popular architectural feature of multi-layer perceptrons. It is generally assumed that by implementing a linear sub-mapping, shortcuts assist the learning process in the remainder of the network. Here we find that this is not always the case: shortcut weights may also act as distractors that slow down convergence and can lead to inferior solutions. This problem can be addressed with *slope centering*, a particular form of *gradient factor centering* [2]. By removing the linear component of the error signal at a hidden node, slope centering effectively decouples that node from the shortcuts that bypass it. This eliminates the possibility of destructive interference from shortcut weights, and thus ensures that the benefits of shortcut connections are fully realized.

1 Shortcuts

Shortcut weights bypass a given hidden node by connecting its inputs directly to the node(s) it projects to. They 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 sub-network 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 node(s) from responsibility for the linear component (now implemented by the shortcuts) of the mapping that it is to learn.

Here we take a closer look at the second argument. It is true that with shortcuts added, a nonlinear network generally attains larger capacity and may therefore be able to better approximate a given mapping. What about the dynamics of gradient descent in such a network though — how do shortcuts affect learning in the bypassed hidden node? Our experiments with single hidden layer networks — where backpropagation through shortcuts does not play a role — suggest that shortcuts actually slow down convergence, and may lead to inferior solutions (see Section 3).

This should not come as a surprise — after all, the simultaneous adaptation of additional parameters (the shortcut weights) at non-infinitesimal step sizes

*Reprinted from *Proc. 8th International Conference on Artificial Neural Networks* [1].

must necessarily add noise to the error signal. The shortcuts add degrees of freedom to which the bypassed hidden nodes are also coupled, and such redundant parametrization impedes optimization by gradient descent. In essence, while bypassed hidden nodes need no longer concern themselves with the linear component of the mapping to be learned, how are they to know this? In order to reap the full benefit of shortcut weights, the linear component must be explicitly subtracted from the error signal for bypassed nodes. With *slope centering* we introduce an efficient and effective way of doing this.

2 Slope Centering

Consider a hidden node with net input y and activation given by $z = f(y)$, where f is a nonlinear (typically sigmoid) function. According to the chain rule, the error gradient with respect to some objective E acquires the factor $f'(y)$ — the node's current *slope* — as it is backpropagated through f:

$$\frac{\partial E}{\partial y} \;=\; f'(y)\,\frac{\partial E}{\partial z} \tag{1}$$

We can split this gradient into two orthogonal components:

$$\frac{\partial E}{\partial y} \;=\; \Big[f'(y) - \big\langle f'(y)\big\rangle\Big]\,\frac{\partial E}{\partial z} \;+\; \big\langle f'(y)\big\rangle\,\frac{\partial E}{\partial z} \tag{2}$$

where $\langle\cdot\rangle$ denotes averaging over input patterns. This average evolves gradually on the slow time scale of the network's weight dynamics; for the pattern-dependent computation of the instantaneous gradient in (2) we can therefore assume $\langle f'(y)\rangle \approx$ const. Note that this means that the second term in (2) is linear: it is the error that would be backpropagated by a linear node in place of the present (nonlinear) hidden node. Conversely, by subtracting this linear error from it, we have made the other (first) term purely nonlinear. We have thus split the hidden node's error signal into its linear and nonlinear components.

As we have argued above, the linear component of the error should be removed for hidden nodes that are bypassed by shortcuts. This leads us to propose backpropagating error signals in that case via

$$\frac{\partial E}{\partial y} \;:=\; \Big[f'(y) - \big\langle f'(y)\big\rangle\Big]\,\frac{\partial E}{\partial z} \tag{3}$$

Since this differs from (1) in that the slope $f'(y)$ has been centered about zero by subtracting its average, we refer to this technique as *slope centering*. Note that (3) does not describe the error gradient proper; it rather expresses that gradient projected into the null space of the shortcut weights.[1] We have thus effectively decoupled the hidden node from its shortcuts; our empirical results (see Section 3) show that such an orthogonalization of parameters can be highly beneficial to the network's learning process.

[1] Rather than introduce additional notation to that effect, we have abused the partial derivative notation here as a placeholder for the slope-centered backpropagated error signal.

In online learning, the average slope $\langle f'(y) \rangle$ in (3) must be approximated, typically by an exponential running average. When batch learning is used, the average slope over the current batch may be approximated by the value computed for the previous batch. Alternatively, one may calculate the slope-centered error (3) for the current batch exactly by accumulating $\sum f'(y)$, $\sum \partial E / \partial z$, and $\sum f'(y)\, \partial E / \partial z$, then using the fact that

$$\left\langle \Big[f'(y) - \langle f'(y) \rangle \Big] \frac{\partial E}{\partial z} \right\rangle \;=\; \left\langle f'(y)\, \frac{\partial E}{\partial z} \right\rangle \;-\; \langle f'(y) \rangle \left\langle \frac{\partial E}{\partial z} \right\rangle \tag{4}$$

3 Empirical Results

We now demonstrate the effect of slope centering on speed and reliability of convergence as well as generalization performance in feedforward networks trained by accelerated gradient descent. After describing the general setup of the experiments, we present our respective results for two benchmarks: the toy problem of symmetry detection in binary patterns, and a difficult vowel recognition task.

3.1 Experimental Setup

For each benchmark we ran four experiments examining the separate and combined effect of shortcuts and slope centering. Each experiment consisted of a number of runs starting from different initial weights drawn from a zero-mean Gaussian distribution with standard deviation 0.3. Training was done in batch mode; the hidden-to-output weights of the network were updated *before* backpropagating error through them [3]. In order to make the optimization as efficient as possible, we tested a number of acceleration methods, then chose the combination of two (*vario-η* and *bold driver*) that yielded the fastest reliable convergence overall.[2] Vario-η [4, 5, page 48] sets the local learning rate for each weight inversely to the standard deviation of its stochastic gradient:

$$\Delta w_{ij} \;=\; \frac{-\eta\, g_{ij}}{\varrho + \sigma(g_{ij})}\,, \quad g_{ij} \;\equiv\; \frac{\partial E}{\partial w_{ij}}\,, \quad \sigma(u) \;\equiv\; \sqrt{\langle u^2 \rangle - \langle u \rangle^2}\,, \tag{5}$$

with the small positive constant $\varrho = 0.1$ preventing division by near-zero values. The global learning rate η was adjusted by the bold driver technique [6, 7, 8, 9]: after each batch in which the error did not increase by more than $\varepsilon = 10^{-10}$ (for numerical stability), η is increased by 2%. Otherwise, the last weight change is undone, and η decreased by 50%. Due to the amount of recomputation they require, we did count those "failed" epochs in our performance figures.

3.2 Symmetry Detection Problem

A fully connected feedforward network with 8 inputs, 8 hidden units (tanh nonlinearity) and a single logistic output is to learn the symmetry detection

[2]Note that any performance advantage for slope centering reported thereafter has thus been realized *on top of* a state-of-the-art accelerated gradient method as control.

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

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

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

task: given a binary pattern (composed of ± 1s) at the input, it is to signal whether the pattern is symmetric about its middle axis (target = 1) or not (target = 0). The network was trained on all 256 patterns, using a cross-entropy loss function and *error centering* [2, 10], until the root-mean-square error of its output fell below 0.01. Since the complement of a symmetric bit pattern is also symmetric, the symmetry detection task has *no* linear component at all — we therefore expected shortcuts to be of minimal benefit here.

Table 1 shows that indeed adding shortcuts alone was not beneficial — it slowed down convergence in over 80 of the 100 runs performed, and significantly increased the c.v. (coefficient of variation). 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 (linear) shortcut weights, the hidden nodes can no longer reduce the linear component of the error signal on their own.

3.3 Vowel Recognition Problem

We also tested our approach on Deterding's speaker-independent vowel recognition data [11], which has been adopted [12] as a popular [13, 14, 15, 16] neural network benchmark. The task is to recognize the eleven steady-state vowels of British English in a speaker-independent fashion, 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 follow the conventional split into training (first 8 speakers) and test set (remaining 7).

We trained fully connected feedforward networks with 10 inputs, 22 logistic hidden units, and 11 logistic output units by minimization of cross-entropy loss. The target was 1 for the output corresponding to the correct vowel, 0 for all others. After each epoch, the network's generalization ability was measured in

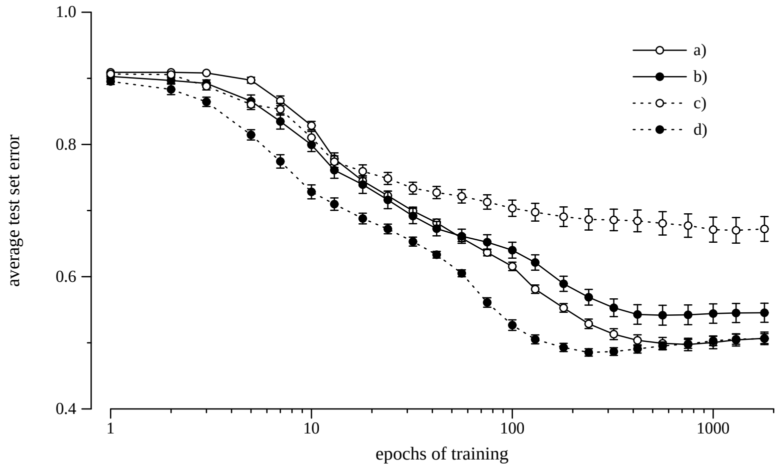

Figure 1: Evolution of the average test set error while learning the vowel recognition task with shortcut weights (filled marks) and/or slope centering (dashed lines). The two techniques worsen performance individually, while their conjunction improves it.

terms of its misclassification rate on the test set. For the purpose of testing, a maximum likelihood approach was adopted: the network's highest output for a given test pattern was taken to indicate its classification of that pattern.

Figure 1 shows how the test set error (averaged over 25 runs) evolved during training. While the addition of shortcut weights alone (b) worsened generalization performance (*cf.* a), in conjunction with slope centering (d) it resulted in faster convergence (by a factor of five), and to lower test test errors. The use of slope centering *without* shortcuts again proved ill-advised (c).

4 Conclusion

We have introduced slope centering, a technique that modifies a hidden node's error signal so as to eliminate interference from shortcut weights that bypass it. Using an already accelerated gradient method as a baseline, we found in two benchmarks that while shortcuts alone worsened performance, their combination with slope centering further sped up learning significantly. Slope centering is one example of the more general *gradient factor centering* approach [2]. It has long been known that centering input and hidden unit activities is beneficial [17, 18], and recently we have extended this notion to the centering of error signals [10]. In future work, we will investigate the centering of additional gradient factors, such as those that occur in networks with multiplicative nodes.

Acknowledgment

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

References

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

[2] N. N. Schraudolph, "Centering neural network gradient factors", In Orr and Müller [19], pp. 207–226, ◁ ftp://ftp.idsia.ch/pub/nic/center.ps.gz ▷.

[3] S. Shah, F. Palmieri, and M. Datum, "Optimal filtering algorithms for fast learning in feedforward neural networks", *Neural Networks*, **5**:779–787, 1992.

[4] R. Neuneier and H. G. Zimmermann, "How to train neural networks", In Orr and Müller [19], pp. 373–423.

[5] H. G. Zimmermann, "Neuronale Netze als Entscheidungskalkül", in *Neuronale Netze in der Ökonomie: Grundlagen und finanzwirtschaftliche Anwendungen*, H. Rehkugler and H. G. Zimmermann, Eds., pp. 1–87. Vahlen Verlag, Munich, 1994.

[6] A. Lapedes and R. Farber, "A self-optimizing, nonsymmetrical neural net for content addressable memory and pattern recognition", *Physica*, **D 22**:247–259, 1986.

[7] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, "Accelerating the convergence of the back-propagation method", *Biological Cybernetics*, **59**:257–263, 1988.

[8] R. Battiti, "Accelerated back-propagation learning: Two optimization methods", *Complex Systems*, **3**:331–342, 1989.

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

[10] N. N. Schraudolph and T. J. Sejnowski, "Tempering backpropagation networks: Not all weights are created equal", in *Advances in Neural Information Processing Systems*, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds. 1996, vol. 8, pp. 563–569, The MIT Press, Cambridge, MA.

[11] D. H. Deterding, *Speaker Normalisation for Automatic Speech Recognition*, PhD thesis, University of Cambridge, 1989.

[12] A. J. Robinson, *Dynamic Error Propagation Networks*, PhD thesis, University of Cambridge, 1989.

[13] M. Finke and K.-R. Müller, "Estimating a-posteriori probabilities using stochastic network models", in *Proceedings of the 1993 Connectionist Models Summer School,*, M. C. Mozer, P. Smolensky, D. S. Touretzky, J. L. Elman, and A. S. Weigend, Eds., Boulder, CO, 1994, Lawrence Erlbaum Associates, Hillsdale, NJ.

[14] M. Herrmann, "On the merits of topography in neural maps", in *Proceedings of the Workshop on Self-Organizing Maps*, T. Kohonen, Ed. Helsinki University of Technology, 1997, pp. 112–117.

[15] S. Hochreiter and J. Schmidhuber, "Unsupervised coding with LOCOCODE", in *Proceedings of the 7th International Conference on Artificial Neural Networks*, Lausanne, Switzerland, 1997, pp. 655–660, Springer Verlag, Berlin.

[16] G. W. Flake, "Square unit augmented, radially extended, multilayer perceptrons", In Orr and Müller [19], pp. 145–163.

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

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

[19] G. B. Orr and K.-R. Müller, Eds., *Neural Networks: Tricks of the Trade*, vol. 1524 of *Lecture Notes in Computer Science*, Springer Verlag, Berlin, 1998.