ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 97-104

Comparison of Optimized Backpropagation Algorithms *

W. Schiffmann, M. Joost, R. Werner
University of Koblenz
Institute of Physics
Rheinau 3–4
W-5400 Koblenz
e-mail: evol@infko.uni-koblenz.de

Abstract

Backpropagation is one of the most famous training algorithms for multilayer perceptrons. Unfortunately it can be very slow for practical applications. Over the last years many improvement strategies have been developed to speed up backpropagation. It is very difficult to compare these different techniques, because most of them have been tested on various specific data sets. Most of the reported results are based on some kind of tiny and artificial training sets like XOR, encoder or decoder. It is very doubtful if these results hold for more complicate practical application. In this report an overview of many different speedup techniques is given. All of them were assessed by a very hard practical classification task, which consists of a big medical data set. As you will see many of these optimized algorithms fail in learning the data set.

1 Introduction

This report is intended to summarize our experience using many different speedup techniques for the backpropagation algorithm. We have tested 16 different algorithms on a very hard classification task. Most of these algorithms are using many parameters, which have to be tuned by hand. So hundreds of tests runs have to be performed. It is beyond the scope of this paper to discuss every approach in detail. We rather group the different approaches in some classes of algorithms and discuss these classes. A more detailed report is available via ftp (128.146.8.52 in /pub/neuroprose/schiff.bp_speedup.ps.Z).

2 Thyroid-Data

In order to compare many different approaches we have used measurements of the thyroid gland [Quinlan, 1987]. Each measurement vector consists of 21 values – 15

*This work is supported by the *Deutsche Forschungsgemeinschaft* (DFG) as part of the project *FE–generator* (grant Schi 304/1–1)

ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 97-104

binary and 6 analog. Three classes are assigned to each of the measurement vectors which correspond to the hyper-, hypo- and normal function of the thyroid gland. Since approximately 92% of all patients have a normal function, a useful classifier must be significantly better than 92% correct classifications. The training set consists of 3772 measurement vectors and again 3428 vectors are available for testing. The training period was limited to 5000 epochs using a fixed 3 layer network architecture with 21 input- , 10 hidden- and 3 output units. The network was fully interconnected. Using a SPARC2 CPU training takes from 12 to 24 hours. The weights of the network have been randomly chosen by a normal distribution ($\mu = 0.0, \sigma = 0.1$). The bias of each unit has been computed as follows. First the average input pattern of the hole learning set has been calculated. While propagating this averaged pattern through the network the bias of each unit is tuned to half activate every hidden or output unit. By this means the gradient of the sigmoid activation function of every unit is maximized, which has some benefits on the gradient descent during the training.

3 Standard Backpropagation

Basically, backpropagation [Rumelhart, 1986] is a gradient descent technique to minimize some error criteria E. In the batched mode variant the descent is based on the gradient ∇E for the total training set :

$$\Delta w_{ij}(n) = -\epsilon * \frac{\partial E}{\partial w_{ij}} + \alpha * \Delta w_{ij}(n-1)$$

ϵ and α are two non-negative constant parameters called learning rate and momentum. The momentum can speed up training in very flat regions of the error surface and suppresses weight oscillation in steep valleys or ravines. Unfortunately it is necessary to propagate the hole training set through the network for calculating ∇E . This can slow down training for bigger training sets. For some tasks (e.g. neural controllers) no finite training set is available. Therefore often a online variant is used, which updates the connections based on the gradient for the actual training pattern ∇E_p:

$$\Delta w_{ij}(n) = -\epsilon * \frac{\partial E_p}{\partial w_{ij}} + \alpha * \Delta w_{ij}(n-1)$$

A good choise of ϵ and α is essential for training success and speed. Adjusting these parameters by hand can be very difficult and might take a very long time for more complicated tasks.

4 Online Training

Only little work is done on improving the online update scheme. Darken and Moody [1990] have developed a so called "Search-Then-Converge" strategey, which simply decreases the learning rate during training. Schmidhuber [1989] calculates a lerarning rate independent for every pattern presentation. Therefore the tangent in the error

surface of the actual training pattern at the current position is used. The new values for every connection are found by calculating the point of intersection with the zero plane. For practical reasons it is necessary to define an upper limit for a single learning step. There also may be some error surfaces which never reach the zero plane. For those surfaces a small constant value is subtracted to make sure that zero points exists.
Figure 1 shows the best training runs of standard online backpropagation, Darken and Moody's approach and Schmidhuber's algorithm. By using a decreasing learning rate during the training the advantages of big values (fast learning in the early learning phase) and small values (good asymptotic behaviour) can be combinded, resulting in slightly improved performance. Unfortunately a new parameter (decay factor) has to be adjusted by hand. Results of Schmidhuber's algorithm are very disappointing and worse than standard online backpropagation.

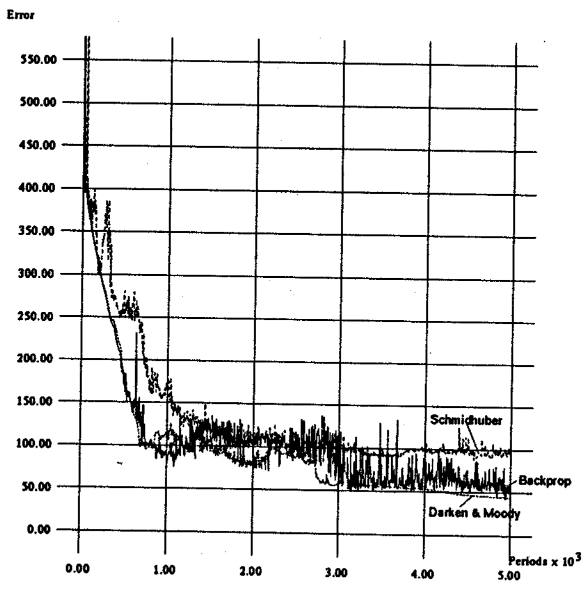

Fig. 1. Online Algorithms

5 Batched Training

Most of the improvement strategies are based on batched backpropagation. Using this update scheme the gradient of the complete error surface is known and can be used to calculate new connection strength more accurately than just using the gradient of the partial error surface given by a single training pattern. Unfortunately much less updates

can be performed, which may slow down the training of big learning sets.
Two different types of batched algorithms can be distinguished. On the one hand there is the global adaptive type, which adapts a single learning rate used for all connections during the training. On the other hand there are local adaptive algorithms, which use independent learning rates for every connection. However, all the learning rates are adapted during the training.

5.1 Global adaptive Algorithms

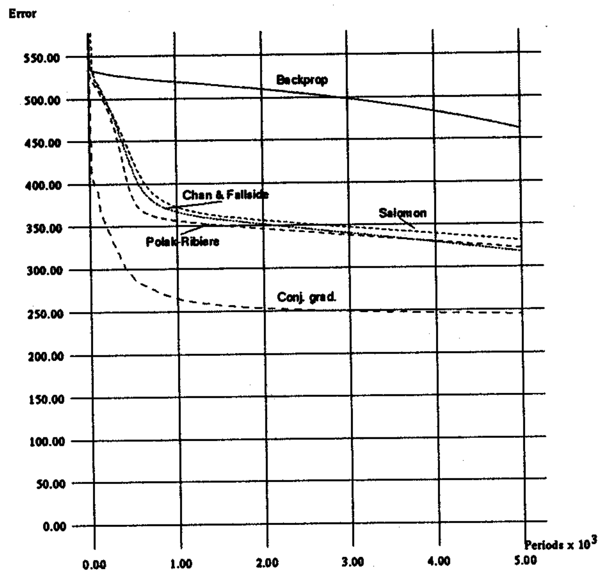

Fig. 2. Global adaptive Algorithms

Many different strategies have been suggested to adapt a global learning rate during the training. Salomon [1989] uses a simple evolution strategy to adjust the learning rate. Therefore starting with some ϵ the next update is done by using an increased and a decreased learning rate. The one which results in better performance is used as the starting point for the next update. Chan and Fallside [1987] adjust the learning rate by observing the angle between $\nabla E(n)$ and $\Delta w(n-1)$. The adaptation tries to adjust this angel at 90^o. As long as the angle is less than 90^o the learning rate is increased otherwise it is decreased. On principle it is possible to calculate the optimal learning rate for an update direction, which minimizes E with respect to the search direction. In order to minimize the number of iterations for this calculation a Newton like iteration like the method of "False Position" [Luenberger, 1973] can be used. This so called "line

ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 97-104

search" is used by A.H. Kramer and Sangiovanni-Vincentelli [1989] together with the Polak-Ribiere method. Leonard and M.A. Kramer [1990] rather combine a conjugate gradient method and a line search strategy.
In oder to compare the results (displayed in Fig. 2) one has to take into account, that the evolution strategy doubles the computation time and the line search typically requires about 3 iterations. This results in trebled computation time. Salomon's , Chan / Fallside's and Kramer / Vincentelli's approach result in almost identical performance, the conjugate gradient method operates a little better. All strategies clearly outperform standard batched backpropagation. Nevertheless standard online backpropagtion is much superior to all these strategies.

5.2 Local adaptive Algorithms

All local adaptive algorithms adjust learning rates for every connection in order to minimize oszillation and to maximize the update step size. Most of them observe the sign of the last two gradients. As long as these gradients agree in sign the corresponding learning rate is increased. If the sign changes an oscillation is detected and the learning rate is decreased. Silva and Almeida [1990] use this strategy. Increasing and decreasing is performed by multiplying the learning rates with some constant values. They also use a total backtracking strategy which restarts an update step if the total error increases by using halfed learning rates. Tollenaere's SuperSAB algorithm [1990] is quite similar to Silva and Almeida's approach. He has modified the update rule such, that updates which result in sign changes of partial derivatives are undone. Due to this kind of partial backtracking no total backtracking is necessary. Additionally an upper limit for the learning rates is used. Jacobs delta-bar-delta algorithm [1988] not only observes the signs of two successive gradients, but rather uses sign changes of an exponential averaged gradient. In contrast to the other algorithms he uses an additive constant in oder to increase the learning rate. Thus he has a linear increase and an exponential decrease. Riedmiller and Braun [1992] are using an adaptive version of the "Manhattan-Learning" called "rprop". The updates are no longer proportional to the partial derivative. They use an independent step size for every connection, which is adjusted like learning rates by the SuperSAB algorithm. In addition a lower limit for these step size is necessary in oder to prevent arithmetic underflow. The update direction depends on the sign of the partial derivative. Also partial backtracking is used. Scott E. Fahlman's quickprop algorithm [1988] applies the method of "False Positon" independent to every connection. Nevertheless the calculated update has to be limited if the step computed by this formula is too large, infinite or uphill on the current gradient. A learning rate is still necessary to start the training or restart it after a 0.0 update. The calculation of the gradient is modified using $o_i * (1 - o_i) + 0.1$ instead of $o_i * (1 - o_i)$ in the computation of the derivative. Also a small weight decay is used.
In addition we have investigated Fahlman and Lebiere's cascaded correlation algorithm [1990]. This algorithm differs in many ways from all other approaches. It begins with a minimal network, then automatically trains and adds new hidden units one by one, creating a multi-layer structure. Weights are adjusted by the quickprop update rule. Since the computation time for a single training epoch depends on the actual network size, it is not very useful to compare this approach in terms of training epochs. So we

rather use the number of computed connections, which can be expressed in terms of training epochs (for a fixed network topology).
Figure 3 shows the results of all local adaptive algorithms. All local adaptive algorithms greatly reduce the training time and improve the network performance. Nevertheless results still differ. RPROP and Quickprop are superior to all other algorithms. SuperSAB's performance is almost similar to Silva and Almeida's approach. Using the delta-bar-delta algorithm it was very hard to find a proper additive increment for the learning rates. We had to use a very small increment in order to ensure convergence, resulting in slower training. The cascaded correlation algorithm outperforms all algorithms using fixed topologies.

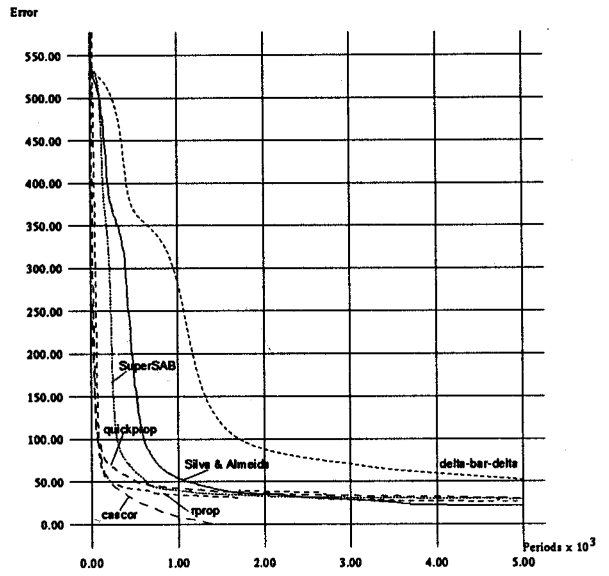

Fig. 3. Local adaptive Algorithms

6 Conclusion

Table 1 shows a comparison of the best results of every training algorithm.

Many "optimized" algorithms failed in training the chosen task, although most authors promised a algorithm superior to standard backpropagation. Because these algorithms have only been tested by training tiny artificial tasks, these results cannot be transfered to more complicated training sets. Especially for these kind of training sets optimization is necessary, whereas it is of little importance to speed up XOR learning. Nevertheless

	Training Set		Testing Set	
Algorithm	Error	Recog. rate	Error	Recog. rate
Backprop	50.3	99.13	137.63	97.58
Backprop (batch mode)	461.8	92.63	414.34	92.85
Backprop (batch mode) + Eaton and Oliver	511.0	92.47	450.10	92.71
Backprop + Darken and Moody	44.2	99.20	126.84	97.90
Schmidhuber	91.5	98.36	163.54	97.23
Salomon	331.1	94.64	346.73	94.14
Chan and Fallside	317.6	94.67	335.47	94.17
Polak-Ribiere + line search	322.0	94.70	339.33	94.17
Conj. gradient + line search	244.9	94.57	267.92	93.84
Silva and Almeida	<u>21.5</u>	<u>99.60</u>	<u>96.65</u>	<u>98.45</u>
SuperSAB	27.9	99.55	105.31	98.42
Delta-Bar-Delta	51.6	99.20	110.64	98.37
RPROP	25.99	99.58	120.73	98.02
Quickprop	29.2	<u>99.60</u>	110.91	98.25
Cascade correlation 10 units	10.37	99.84	91.50	98.42
Cascade correlation 20 units	0.82	100.00	101.36	98.48

Table 1. Best results

most of the algorithms are superior to standard backpropagation running in batched mode. On the other hand online backpropagation outperforms all global adaptive learning algorithms. Algorithms using local adaptation strategies greatly reduce the training time and also improve the network performance. In terms of learning speed RPROP and Quickprop seem to be superior to all other training algorithms using fixed topologies . Nevertheless Silva and Almeida's approach and SuperSAB have trained networks, which generalize a little better. The cascade correlation algorithm clearly outperforms all other approaches but it is not directly comparable with them.

Most algorithms are using batched updates. Very little optimization is done on backpropagation updating connections with respect to the actual training pattern. Further research in needed on this topic. There seems to be little influence on the generalization ability. Nevertheless a network's generalization ability depends on the network topology, as the cascade correlation algorithm shows. Training a network with more and more hidden units just increases the approximation quality with respect to the learning set but does not improve the generalization behaviour.

7 References

1. Chan, L. W. and Fallside, F.: *An adaptive training algorithm for backpropagation networks*, Computer Speech and Language, Vol. 2, page 205-218,1987
2. Darken, C. and Moody, J.: *Note on Learning Rate Schedules for Stochastic Optimization*, Neural Information Processing Systems , Lippmann R. P. and Moody J. E. and Touretzky D. S. (Editors), page 832-838, 1991

3. Fahlman, Scott E.: *An Empirical Study of Learning Speed in Back-Propagation Networks*, Technical Report CMU-CS-88-162, 1988

4. Fahlman, Scott E. and Lebiere, Christian: *The Cascade-Correlation Learning Architecture*, Neural Information Processing Systems 2, page 524-532, 1990

5. Hertz, John and Krogh, Anders and Palmer, Richard G.: *Introduction to theory of neural computation*, Addison-Wesley Publishing Company, ISBN 0-201-51560-1, 1991

6. Jacobs, Robert A.: *Increased Rates of Convergence Through Learning Rate Adaption*, Neural Networks, Vol. 1, page 295-307,1988

7. Joost, Merten and Werner, Randolf: *Algorithmen zur Optimierung neuronaler Merkmalsfilter*, Diplomarbeit, Universität Koblenz, 1991

8. Kramer, Alan H. and Vincentelli, A. Sangiovanni: *Efficient parallel learning algorithms for neural networks*, Advances in Neural Information Processing Systems I, Touretzky D. S. (Editor), page 40-48, 1989

9. Leonard, J. and Kramer, M. A.:*Improvement of the Backpropagation Algorithm for Training Neural Networks*, Computers Chem. Engng., Volume 14, No 3, page 337-341, 1990

10. Luenberger, David G.: *Introduction to linear and nonlinear programming*, Addison-Wesley, 1973

11. Quinlan J.R.: *Simplifying decision trees*, Int. J. Man-Machine Studies, page 221-234, 1987

12. Rumelhart D.E., Hinton G.E. and Williams R.J., 1986: *Learning internal representations by error propagation*, in Parallel Distributed Processing: *Explorations in the Microstructures of Cognition*, Vol.I, MIT Press, pp. 318–362

13. Riedmiller, Martin and Braun, Heinrich: *RPROP - A Fast Adaptive Learning Algorithm*, Technical Report (To appear in: Proc. of ISCIS VII), Universität Karlsruhe, 1992

14. Salomon, Ralf: *Adaptive Regelung der Lernrate bei back-propagation*, Forschungsberichte des Fachbereichs Informatik, Technische Universität Berlin, Bericht 1989/24, 1989

15. Scalero, Robert S. and Tepedelenlioglu, Nazif: *A Fast Algorithm for Training Feedforward Neural Networks*, IEEE Transactions on Signal Processing, Vol. 40, No 1, page 202-210, January 1992

16. Schmidhuber, Jürgen: *Accelerated Learning in Back-Propagation Nets*, Connectionism in perspective, Elsevier Science Publishers B.V. (North-Holland), page 439-445, 1989

17. Silva, Fernando M. and Almeida, Luis B.: *Speeding up Backpropagation*, Advanced Neural Computers, Eckmiller R. (Editor), page 151-158, 1990

18. Tollenaere, Tom: *SuperSAB: Fast Adaptive Backpropagation with Good Scaling Properties*, Neural Networks, Vol. 3, page 561-573, 1990