An Online Backpropagation Algorithm with Validation Error-Based Adaptive Learning Rate

Stefan Duffner and Christophe Garcia

Orange Labs, 4, Rue du Clos Courtel, 35512 Cesson-Sévigné, France
{stefan.duffner,christophe.garcia}@orange-ftgroup.com

Abstract. We present a new learning algorithm for feed-forward neural networks based on the standard Backpropagation method using an adaptive global learning rate. The adaption is based on the evolution of the error criteria but in contrast to most other approaches, our method uses the error measured on the *validation* set instead of the *training* set to dynamically adjust the global learning rate. At no time the examples of the validation set are directly used for training the network in order to maintain its original purpose of validating the training and to perform "early stopping". The proposed algorithm is a heuristic method consisting of two phases. In the first phase the learning rate is adjusted after each iteration such that a minimum of the error criteria on the validation set is quickly attained. In the second phase, this search is refined by repeatedly reverting to previous weight configurations and decreasing the global learning rate. We experimentally show that the proposed method rapidly converges and that it outperforms standard Backpropagation in terms of generalization when the size of the training set is reduced.

1 Introduction

The Backpropagation (BP) algorithm [1] is probably the most popular learning algorithm for multilayer perceptron (MLP)-type neural architectures due to its simplicity and effectiveness. However, the choice of the learning rate used when updating the weights is crucial for the successful convergence and the generalization capacity of the network. A too small learning rate leads to slow convergence and a too high learning rate to divergence. Moreover, in the latter case the network is likely to overfit to the training data when using an *online* Backpropagation algorithm as it might specialize to the examples presented at the beginning of the training. Numerous solutions for the dynamic adaptation of the learning rate have been proposed in the literature. Most of them focus on the acceleration of the training process rather than their generalization performance. They can roughly be divided into two groups: global and local adaption techniques. The former is referring to methods adjusting an overall learning rate for the whole network and the latter to the adaptation of independent learning rates for each weight.

A method for global adaptation has been proposed by Chan et al. [2] where the angle between the last weight update and the current gradient is calculated. If it

J. Marques de Sá et al. (Eds.): ICANN 2007, Part I, LNCS 4668, pp. 249–258, 2007.

is less than 90° the learning rate is increased otherwise it is decreased. Salomon et al. [3] proposed an evolutionary based adaption of the learning rate. At each iteration, two weight updates, one with increased and with decreased learning rate, are performed separately. The resulting network that performs better is retained and used as a starting point for the next iteration. A heuristic method, the so-called "bold driver" method, has been employed by Battiti et al. [4] and Vogl et al. [5]. Here the learning rate is adjusted according to the evolution of the error criteria E. If E decreases the learning rate is slightly increased, otherwise it is drastically decreased. Hsin et al. [6] propose to use a weighted average of the cosines between successive weight updates, and Plagianakos et al. [7] calculate a two point approximation to the secant equation underlying quasi-Newton methods in order to obtain a dynamic learning rate and additionally make use of an acceptability condition to ensure convergence. LeCun et al. [8] calculate a global learning rate by an online estimation of the largest eigenvalue of the Hessian matrix. They show that the optimal learning rate is approximately the inverse of this largest eigenvalue. Finally, the approach of Magoulas et al. [9] estimate the local shape of the error surface by the Lipschitz constant and set the learning rate accordingly. They also applied this technique to calculate a separate dynamic learning rate for each weight [10].

Local learning rate adjustment methods have been very popular due to their efficiency and generally higher convergence speed. A very well-know technique is the Delta-Bar-Delta method introduces by Jacobs et al. [11]. Here, the learning rates are adjusted according to sign changes of the exponential averaged gradient. Similarly, Silva and Almeida [12] proposed a method where the learning rates are increased if the respective gradients of the last two iterations have the same size and decreased otherwise. The RPROP method introduced by Riedmiller et al. [13] uses a step size which doesn't depend on the gradient magnitude but which is increased or decreased according to gradient sign changes.

Finally, many methods do not use an explicit learning rate but first calculate a descent gradient direction and then perform a line search such that the error criteria is minimized in the direction of the gradient [14, 15, 16].

Note that most of the existing adaptive learning algorithms are *batch* learning algorithms, i.e. the weights are updated after all examples have been presented to the network. On the other hand, *online* algorithms update the weights after the presentation of each example. They generally converge faster when the input space is large compared to the number of examples (e.g. in image processing tasks) or in more complex architectures like convolutional neural networks (CNN) that use shared weights. Thus, for many real world applications the online Backpropagation algorithm or its variants are still the best choice. There are also some adaptive online algorithms in the literature. For example Schraudolph [17], Harmon et al. [18] and Almeida et al. [19] proposed methods similar to the Incremental Delta-Bar-Delta approach introduced by Sutton et al. [20], an extension of the Delta-Bar-Delta technique for stochastic training. However, these algorithms mainly aim at a faster convergence rather than an increased generalization capacity.

We present a heuristic learning algorithm that improves generalization capacity and shows good convergence speed. It is an online Backpropagation method with adaptive global learning rate. The learning rate adaption is based on the idea of the so-called "bold driver" method [5, 4], i.e. the learning rate is initialised with a very small value and increased or decreased according to the evolution of the error criteria E of the past iterations. The main difference with respect to previous works is that the error is not measured on the *training* but on the *validation* set. Some further optimizations have been made in order to ensure fast convergence. The aim of this heuristic approach is not only to accelerate convergence compared to standard online Backpropagation but also to improve generalization.

The remainder of this paper is organized as follows: Section 2 describes the training algorithm with its two phases. In section 3, experimental results are presented and finally in section 4 we draw our conclusions.

2 The Training Algorithm

The proposed training method is an extension of the standard Backpropagation algorithm [1]. Backpropagation is a gradient descent technique that minimizes an error criteria E which is usually the mean squared error (MSE) of the N output values o_{ij} with respect to its desired values d_{ij} of the neural network:

$$E = \frac{1}{NP} \sum_{j=1}^{P} \sum_{i=1}^{N} (o_{ij} - d_{ij})^2 \quad , \tag{1}$$

where P is the number of examples in the training set.

Training is performed *online*, i.e. the weights of the neural network are updated after presentation of each training example. Classically, at iteration t a given weight $w_{ij}(t)$ is updated by adding a $\Delta w_{ij}(t)$ to it:

$$\Delta w_{ij}(t) = -\epsilon(t) \cdot \frac{\partial E(t)}{\partial w_{ij}(t)} \quad , \tag{2}$$

where ϵ is the learning rate.

The training database is usually divided into a *training set* and a *validation set*. The training, i.e. the Backpropagation, is only performed on the training set. The purpose of the validation set is to determine when to stop training in order to obtain a neural network that generalizes sufficiently well. This technique is commonly known as "early stopping". In the proposed algorithm the validation set has a second role. It is used to control the adjustment of the learning rate $\epsilon(t)$ after each training iteration t. Note, that the learning rate adjustment is performed only once per iteration. Our approach basically consists of two phases:

1. the main learning phase and
2. the refinement phase.

In the following sections we will detail these steps.

2.1 The Main Learning Phase

The adaption of the learning rate in our approach is similar to the "bold driver" method [5, 4] where the learning rate is initialized with a very small value (e.g. 10^{-10}) and adjusted after each training iteration according the difference of the error criteria E between the current and the preceding iteration.

The proposed method applies this idea to the validation set instead of the training set in order to reduce overfitting. Moreover, the procedure is slightly modified to be more tolerant to error increases as the validation error is more likely to oscillate than the training error. Let us consider the typical runs of the error curves of a training and validation set when using standard Backpropagation. Fig. 1 illustrates this in a simplified manner. When applying the technique

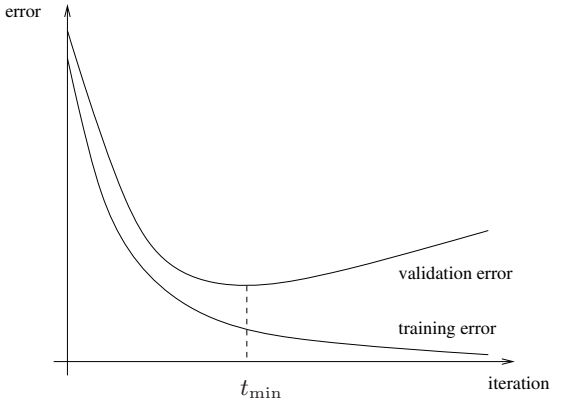


Fig. 1. The typical evolution of the error criteria evaluated on the training and validation set

of "early stopping" the weight configuration at iteration $t_{\min}$, i.e. where the validation error is minimal, is retained as the network is supposed to show the highest generalization performance at this point. Further training likely leads to overfitting. The purpose of the first phase of the proposed algorithm is thus to reach the point $t_{\min}$ more quickly.

To this end, the normalized difference between the error criteria of the current and the preceding iteration is calculated:

$$\delta(t) = \frac{E_v(t) - E_v(t-1)}{E_v(t)} \quad . \tag{3}$$

$E_v(t)$ is the error criteria at iteration t calculated on the whole validation set (cf. Eqn. 1).

The algorithm further requires a running average $\bar{\delta}(t)$ of the preceding values of δ:

$$\bar{\delta}(t) = \alpha \cdot \delta(t) + (1-\alpha) \cdot \bar{\delta}(t-1) \quad , \tag{4}$$

where $0 < \alpha \leq 1$ (e.g. $\alpha = 0.1$).

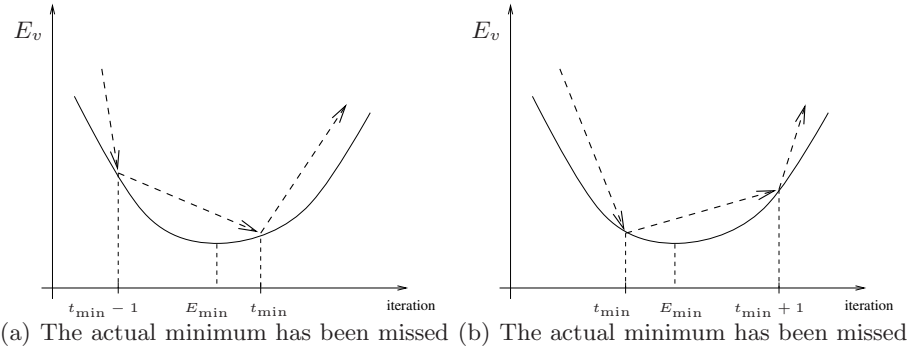
The principal learning rate updating rule is the following:

$$\epsilon(t) = \begin{cases} d \cdot \epsilon(t-1) & \text{if } \delta(t) \cdot \bar{\delta}(t-1) < 0 \text{ and } |\bar{\delta}(t-1)| > \theta \,, \\ u \cdot \epsilon(t-1) & \text{otherwise} \,. \end{cases} \tag{5}$$

where u and d are positive constants, $0 < d < 1 < u$, and θ is a threshold to allow for small error oscillations. In our experiments we used $u = 1.1$, $d = 0.5$ and $\theta = 0.01$. Thus, the learning rate adaption is based on the signs of the error differences of the current and the preceding iterations. If the sign changes the learning rate is decreased otherwise it is increased. The principle of this procedure is similar to the Delta-Bar-Delta method [11] but the calculation is not based on gradients.

2.2 Refinement Phase

If the training has passed iteration $t_{\min}$ where E_v is minimal the network is likely to overtrain. In fact, as the gradient descent is performed in discrete steps the actual minimum $E_{\min}$ of the error surface of the validation set is likely to be missed and lies between the weight configurations of two successive training iterations. Fig. 2 illustrates this. Clearly, there are two cases to differentiate:



(a) The actual minimum has been missed *before* iteration $t_{\min}$

(b) The actual minimum has been missed *after* iteration $t_{\min}$

Fig. 2. The two possible cases that can occur when the minimum on the validation set is reached

either the minimum $E_{\min}$ has been missed before or after iteration $t_{\min}$. Now we assume that the validation error surface is relatively smooth and that no other local minimum lies between iterations $t_{\min} - 1$ and $t_{\min}$ or between iterations $t_{\min}$ and $t_{\min} + 1$ respectively. In order to try to attain a smaller error the network reverts to the weight configuration at iteration $t_{\min} - 1$, decreases the learning rate and training is continued. Note that for training only the examples of the training set are used. Thus, it is uncertain if the actual minimum can be attained at all. If no smaller error has been found for a certain number of iterations T the "real" minimum is more likely to have occurred "after" iteration

$t_{\min}$, (see Fig. 2(b)). In this case, the network reverts to iteration $t_{\min}$, the learning rate is again decreased and training continues. If a smaller error is reached during this process the temporary minimum is retained and the training continues normally. Otherwise the reverting procedure is repeated while always retaining the absolute minimum and the respective weight configuration found so far. Algorithm 1 summarizes the overall training procedure. Note that the computational overhead of the algorithm compared to standard backpropagation with fixed learning rate is negligible as the error on the validation set needs to be calculated anyway to do the "early stopping".

Fig.3 illustrates a typical evolution of the error criteria $E_v(t)$ during the training process using the proposed learning algorithm. Because of the initialization with a very small value the error stays nearly constant at the beginning but drops very quickly at some point due to the exponential increase of the learning rate, and finally it converges to a minimum. In general, the main part of the minimization is done in the first phase and the error decrease in the refinement phase is relatively small.

Algorithm 1. The basic learning algorithm

Initialize weights and individual learning rates
Set $\epsilon(0) := 10^{-10}$, $\delta(0) := 0$ and $\bar{\delta}(0) := 0$
Calculate $E_v(0)$
$t := 0$
repeat
  Do one training iteration
  $t := t + 1$
  Calculate $\delta(t) = \frac{E_v(t) - E_v(t-1)}{E_v(t)}$
  if $\delta(t) \cdot \bar{\delta}(t-1) < 0$ and $|\bar{\delta}(t-1)| > \theta$ **then**
    $\epsilon(t) = d \cdot \epsilon(t-1)$
  else
    $\epsilon(t) = u \cdot \epsilon(t-1)$
  end if
  $\bar{\delta}(t) = \alpha \cdot \delta(t) + (1-\alpha) \cdot \bar{\delta}(t-1)$
  if $E_v(t) < E_{\min}(t)$ **then**
    save the current weight configuration
    $t_{\min} := t$
  end if
  if $t - t_{\min} > T$ **then**
    Revert to weight configuration at $t_{\min} - 1$ (or $t_{\min}$)
  end if
until $t = t_{\max}$

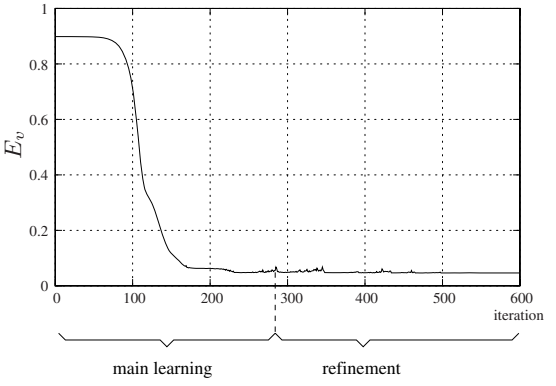


Fig. 3. A typical evolution of the error criteria on the validation set using the proposed learning algorithm

3 Experimental Results

We evaluated the proposed learning algorithm on a MLP trained to classify the examples of the well-known NIST database of handwritten digits. The database contains 3823 training and 1797 test examples of 8x8 matrices. From the training set 500 examples were selected randomly and used for validation. The MLP we used for the experiments had 64 input, 10 hidden and 10 output neurons, fully inter-connected. The neurons all had sigmoid activation functions.

To ensure that the neural network is well-conditioned we additionally use *fixed local* learning rates that are distributed stepwise from the last layer to the first layer according to the incoming connections of each neuron. Thus, the output neuron(s) have the highest and the input the lowest local learning rate. The overall learning rate is just the product of the fixed local and the dynamic global learning rate.

In the first experiment, we compare the convergence properties of the proposed algorithm to the ones of standard Backpropagation. Fig. 4 shows the resulting error curves evaluated on the validation set. The different curves for the Backpropagation algorithm have been obtained by using different global learning rates ($10^{-3}, 10^{-4}$ and 10^{-5}). The global learning rate of the proposed method was initialized with the value 10^{-7}. Note that our approach converges more slowly at the beginning but catches up quickly and finishes stable on the same level or even lower than Backpropagation.

Fig. 5 illustrates 0that our method is not sensitive to different initializations of the global learning rate. The curves show the validation error curves for three different runs with initial learning rates of 10^{-6}, 10^{-8} and 10^{-10} respectively. Note that the point of time where the minimum is reach increases only *linearly* when the initial learning rate is decreased *exponentially*. This is another side effect of the exponential learning rate update rule. All the runs converge to approximately the same solution, and the recognition rates are about the same for all networks.

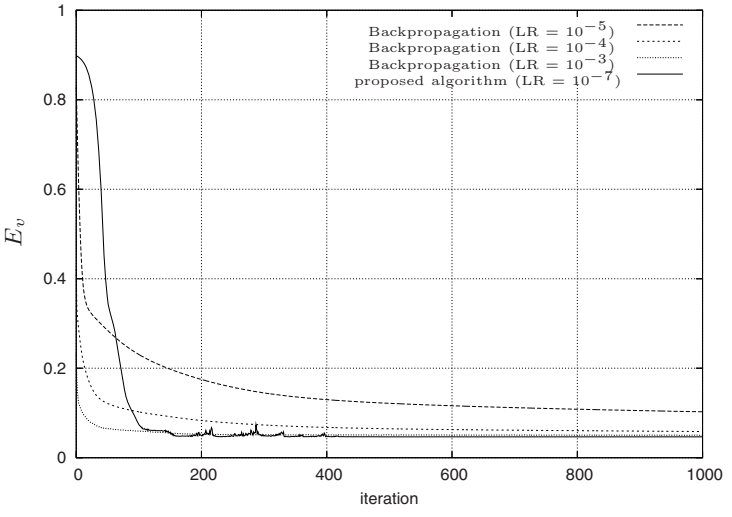


Fig. 4. The evolution of the validation error on the NIST database using Backpropagation and the proposed algorithm

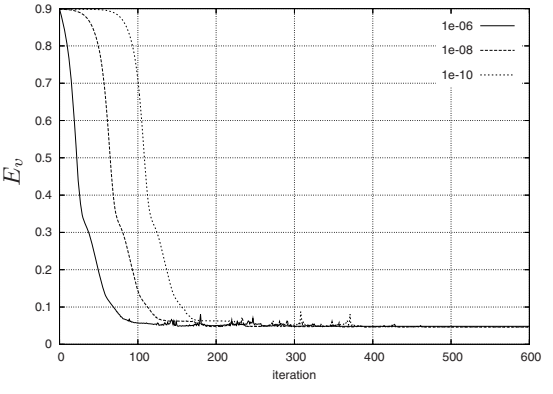


Fig. 5. The validation error curves with different initial global learning rates

The final experiment demonstrates that the algorithm not only converges faster but also improves the generalization performance of the resulting neural networks. To this end the training set was gradually reduced and the respective recognition rates on the test set were calculated and compared to the standard Backpropagation as well as to the bold driver method [5, 4]. Table 1 shows the overall results. One can see that the proposed method performs slightly better with training set sizes 3323 and 1000 and clearly outperforms the other algorithms when only 600 and 100 training examples are used.

Table 2 shows the respective results with a neural network with 40 hidden neurons. The recognition rates of the proposed method are slightly better then

Table 1. Recognition rate (in %) with varying training set size (10 hidden neurons)

training set size / algorithm	3323	1000	600	100
Backpropagation	94.30	93.42	78.31	73.88
bold driver [5, 4]	93.41	91.32	83.74	72.75
proposed algorithm	94.50	93.71	**85.29**	**78.10**

Table 2. Recognition rate (in %) with varying training set size (40 hidden neurons)

training set size / algorithm	3323	1000	600	100
Backpropagation	95.71	93.89	86.58	80.31
bold driver [5, 4]	94.97	93.20	86.45	79.96
proposed algorithm	95.77	93.81	87.06	80.47

for the other algorithms albeit the difference is less significant. However, convergence speed is still superior as illustrated in Fig. 4.

4 Conclusion

We presented a learning algorithm with adaptive global learning rate based on the online Backpropagation method. The adaption is performed in two successive phases, the main learning and a refinement phase. In the main learning phase the learning rate is increased or decreased according to the evolution of the validation error. This leads to a fast and robust convergence to the minimum where "early stopping" is performed. In the second phase the network reverts to preceding weight configurations in order to find a minimum close to the one found in the first step. We experimentally show that this method converges faster than Backpropagation while exhibiting a superior generalization capacity.

References

[1] Rumelhart, D.E., McClelland, J.L.: Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Foundations, vol. 1. MIT Press, Cambridge, MA, USA (1986)
[2] Chan, L.W., Fallside, F.: An adaptive learning algorithm for backpropagation networks. Computer Speech and Language 2, 205–218 (1987)
[3] Salomon, R.: Improved convergence rate of back-propagation with dynamic adaptation of the learning rate. In: Schwefel, H.-P., Männer, R. (eds.) Parallel Problem Solving from Nature. LNCS, vol. 496, pp. 269–273. Springer, Heidelberg (1991)
[4] Battiti, R.: Accelerated backpropagation learning: Two optimization methods. Complex Systems 3, 331–342 (1989)

[5] Vogl, T., Mangis, J., Rigler, J., Zink, W., Alkon, D.: Accelerating the convergence of the back-propagation method. Biological Cybernetics 59, 257–263 (1988)
[6] Hsin, H.C., Li, C.C., Sun, M., Sclabassi, R.: An adaptive training algorithm for back-propagation neural networks. In: International Conference on Systems, Man and Cybernetics, vol. 2, pp. 1049–1052 (1992)
[7] Palgianakos, V., Vrahatis, M., Magoulas, G.: Nonmonotone methods for backpropagation training with adaptive learning rate. In: International Joint Conference on Neural Networks, vol. 3, pp. 1762–1767 (1999)
[8] LeCun, Y., Simard, P., Pearlmutter, B.: Automatic learning rate maximization by on-line estimation of the hessian's eigenvectors. In: Hanson, S., Cowan, J., Giles, L. (eds.) Advances in Neural Information Processing Systems, vol. 5, Morgan Kaufmann Publishers, San Mateo, CA (1993)
[9] Magoulas, G.D., Vrahatis, M.N., Androulakis, G.S.: Effective back-propagation with variable stepsize. Neural Networks 10, 69–82 (1997)
[10] Magoulas, G.D., Vrahatis, M.N., Androulakis, G.S.: Improving the convergence of the backpropagation algorithm using learning rate adaptation methods. Neural Computation 11(7), 1769–1796 (1999)
[11] Jacobs, R.A.: Increased rates of convergence through learning rate adaption. Neural Networks 1, 295–307 (1988)
[12] Silva, F.M., Almeida, L.B.: Acceleration techniques for the backpropagation algorithm. In: Proceedings of the EURASIP Workshop 1990 on Neural Networks, London, UK, pp. 110–119. Springer, Heidelberg (1990)
[13] Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: Proc. of the IEEE Intl. Conf. on Neural Networks, San Francisco, CA, pp. 586–591. IEEE Computer Society Press, Los Alamitos (1993)
[14] Luenberger, D.G.: Introduction to linear and nonlinear programming. Addison-Wesley, Reading (1973)
[15] Fahlman, S.E.: Faster-learning variations on back-propagation: An empirical study. In: Touretzky, D.S., Hinton, G.E., Sejnowski, T.J. (eds.) Connectionist Models Summer School, pp. 38–51. Morgan Kaufmann Publishers, San Mateo, CA (1988)
[16] Leonard, J., Kramer, M.: Improvement of the backpropagation algorithm for training neural networks. Computers & Chemical Engineering 14(3), 337–341 (1990)
[17] Schraudolph, N.: Local gain adaptation in stochastic gradient descent. In: ICANN. Ninth International Conference on Artificial Neural Networks, vol. 2, pp. 569–574 (1999)
[18] Harmon, M., Baird III, L.: Multi-player residual advantage learning with general function approximation. Technical report, Wright Laboratory, WL/AACF, Wright-Patterson Air Force Base, OH (1996)
[19] Almeida, L.B., Langlois, T., Amaral, J.D., Plakhov, A.: Parameter adaptation in stochastic optimization. In: Saad, D. (ed.) On-Line Learning in Neural Networks, Cambridge University Press, Cambridge (1999)
[20] Sutton, R.S.: Adapting bias by gradient descent: an incremental version of beltabar-delta. In: Proceedings of the Tenth International Conference on Machine Learning, Cambridge, MA, pp. 171–176 (1992)