ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

A Regularized Learning Method for Neural Networks Based on Sensitivity Analysis

B. Guijarro-Berdiñas, O. Fontenla-Romero,
B. Pérez-Sánchez and A. Alonso-Betanzos *

Department of Computer Science, University of A Coruña, Spain
(e-mail: {cibertha, ofontenla, bperezs, ciamparo}@udc.es)

Abstract. The Sensitivity-Based Linear Learning Method (SBLLM) is a learning method for two-layer feedforward neural networks, based on sensitivity analysis, that calculates the weights by solving a system of linear equations. Therefore, there is an important saving in computational time which significantly enhances the behavior of this method compared to other learning algorithms. This paper introduces a generalization of the SBLLM by adding a regularization term in the cost function. The theoretical basis for the method is given and its performance is illustrated.

1 Introduction

There exists many successful algorithms for training feedforward neural networks. Most of them are based on the classical gradient descent method [1]. For this reason, many of these algorithms have two main drawbacks: convergence to local minima and slow learning speed. In [2] the authors described a new learning method, Sensitivity-Based Linear Learning Method (SBLLM), that contributes to the solution of the second problem. The main innovations of this method are: 1) the cost function that is minimized is based on the sensitivities of each layer's parameters with respect to its inputs and outputs and 2) the weights of each layer of the network are calculated by solving a system of linear equations. As a result, the method exhibits a higher speed and reaches a minimum error in few epochs of training [2]. This behavior is very suitable in situations that deal with large data sets and networks. However, when the training set is not representative enough, the few iterations employed by the method makes it very difficult to avoid overtraining just employing techniques like *early stopping*.

An usual technique to avoid overfitting is regularization that consists in adding a penalty term to the loss function. In this paper, a generalized version of the SBLLM is presented that includes a regularization term based on the well-known *weight decay* regularizer [3] defined as the sum of squares of all adaptive parameters in the network.

*We would like to acknowledge support from the Xunta de Galicia (PGIDT05TIC10502PR) and the Ministerio de Educación y Ciencia, Spain (TIN2006-02402). Also, we thank the Supercomputing Center of Galicia (CESGA) for allowing us the use of the high performance computing servers.

ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

2 The Sensitivity Learning Algorithm with regularization

Consider the network in Fig. 1. Being S the size of the training set, with I inputs, x_{is}, and J desired outputs, d_{js}, the proposed method is described.

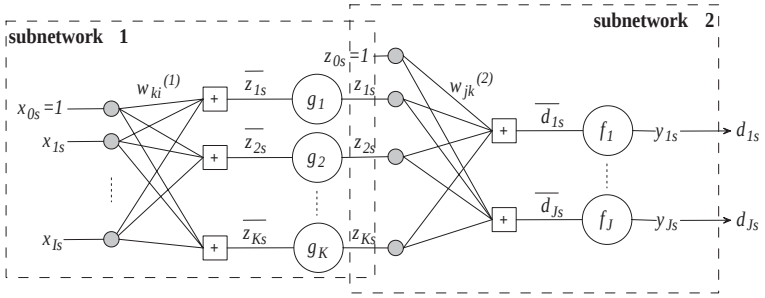

Fig. 1: Two-layer feedforward neural network.

Step 0: Initialization. Initialize the outputs of the intermediate layer as:

$$z_{ks} = g_k\left(\sum_{i=0}^{I} w_{ki}^{(1)}(0)x_{is}\right) + \epsilon_{ks}; \quad \epsilon_{ks} \sim U(-\eta,\eta); k = 1,\ldots,K; s = 1,\ldots,S,$$

where η is a small number, U is an uniform distribution and $\mathbf{w}^{(1)}(0)$ are some random weights.

Step 1: Subproblem solution. Consider the network as composed of two subnetworks (see Fig. 1). The weights of layers 1 and 2 are calculated *independently* by minimizing *for each* layer l a loss function:

$$Q^{(l)} = L^{(l)} + \alpha \sum_{i=0}^{M}\sum_{j=0}^{N} w_{ji}^2, \tag{1}$$

where α is the regularization parameter, the second term on the right-side hand is the regularization term, and M and N are the number of inputs and outputs of layer l. The term $L^{(l)}$ measures the training error. In this work, we consider $L^{(l)}$ as the sum of squared errors *before* the nonlinear activation functions [4, 5]. Therefore the loss function used for solving subnetwork 1 can be written as

$$L^{(1)} = \sum_{s=1}^{S}\sum_{k=1}^{K}(g_k'(\bar{z}_{ks})\bar{\varepsilon}_{ks})^2 = \sum_{s=1}^{S}\sum_{k=1}^{K}\left(g_k'(\bar{z}_{ks})\left(\sum_{i=0}^{I} w_{ki}^{(1)}x_{is} - \bar{z}_{ks}\right)\right)^2 \tag{2}$$

where z_{ks} is the desired output for hidden neuron k, $g_k'(\bar{z}_{ks})$ is the derivative of the activation function and $\bar{z}_{ks} = g_k^{-1}(z_{ks})$. Analogously is defined the loss function for subnetwork 2:

$$L^{(2)} = \sum_{s=1}^{S}\sum_{j=1}^{J}(f_j'(\bar{d}_{js})\bar{\varepsilon}_{js})^2 = \sum_{s=1}^{S}\sum_{j=1}^{J}\left(f_j'(\bar{d}_{js})\left(\sum_{k=0}^{K} w_{jk}^{(2)}z_{ks} - \bar{d}_{js}\right)\right)^2, \tag{3}$$

where d_{js} is the desired output for output neuron j and $\bar{d}_{js} = f_j^{-1}(d_{js})$. The terms $g_j'(\bar{z}_{ks})$ in eq. 2 and $f_j'(\bar{d}_{js})$ in eq. 3 are scaling terms which weigh the errors to ensure that they are magnified appropriately according to the operation point of the nonlinearity at the corresponding value of the desired data point. The inclusion of this term in the loss functions in eq. 2 and 3 is another improvement with respect to the original SBLLM [2]. The advantage of these loss functions is that the optimum set of weights, can be easily calculated by solving the systems of linear equations that are obtained by calculating derivative of $Q^{(1)}$ and $Q^{(2)}$ with respect to the weights and equating to zero:

$$\sum_{i=0}^{I} A_{pi}^{(1)} w_{ki}^{(1)} + \alpha w_{kp}^{(1)} = b_{pk}^{(1)}; \quad p = 0, 1, \ldots, I; \quad k = 1, \ldots, K$$

$$\sum_{k=0}^{K} A_{qk}^{(2)} w_{jk}^{(2)} + \alpha w_{jq}^{(2)} = b_{qj}^{(2)}; \quad q = 0, 1, \ldots, K; \quad j = 1, \ldots, J,$$

where $A_{pi}^{(1)} = \sum_{s=1}^{S} x_{is} x_{ps} g_k'^2(\bar{z}_{ks})$; $b_{pk}^{(1)} = \sum_{s=1}^{S} \bar{z}_{ks} x_{ps} g_k'^2(\bar{z}_{ks})$; $\bar{z}_{ks} = g_k^{-1}(z_{ks})$ and $A_{qk}^{(2)} = \sum_{s=1}^{S} z_{ks} z_{qs} f_j'^2(\bar{d}_{js})$; $b_{qj}^{(2)} = \sum_{s=1}^{S} \bar{d}_{js} z_{qs} f_j'^2(\bar{d}_{js})$; $\bar{d}_{js} = f_j^{-1}(d_{js})$.

Step 2: Evaluate the sum of squared errors. Using the new weights, the MSE is evaluated for the entire network and also a new cost function defined as $Q(\mathbf{z}) = Q^{(1)}(\mathbf{z}) + Q^{(2)}(\mathbf{z})$. This cost function measures the global errors of the network as the sum of the errors of each layer but *before* the nonlinearities, as opposed to the MSE. Later on, based on this cost function, the new values of the $\mathbf{z}$ will be obtained.

Step 3: Convergence checking. If $|Q - Q_{previous}| < \gamma$ or $|MSE_{previous} - MSE| < \gamma'$ stop and return the weights and the sensitivities.

Step 4: Check improvement of Q. If $Q > Q_{previous}$ reduce the value of the step size (ρ), and restore the weights, $\mathbf{z} = \mathbf{z}_{previous}$, $Q = Q_{previous}$ and go to Step 5. Otherwise, store the values $Q_{previous} = Q$, $MSE_{previous} = MSE$ and $\mathbf{z}_{previous} = \mathbf{z}$ and obtain the sensitivities of the cost function Q respect to the output $\mathbf{z}$ of the hidden layer, $\dfrac{\partial Q}{\partial z_{ks}} = \dfrac{\partial Q^{(1)}}{\partial z_{ks}} + \dfrac{\partial Q^{(2)}}{\partial z_{ks}}$ where, $\dfrac{\partial Q^{(1)}}{\partial z_{ks}}$ is defined as,

$$-2\left(g_k^{'}(\bar{z}_{ks})(g^{-1})^{'}(z_{ks})\left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - \bar{z}_{ks}\right)\right)\left(g_k^{'}(\bar{z}_{ks}) - g_k^{''}(\bar{z}_{ks})\left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - \bar{z}_{ks}\right)\right)$$

and

$$\frac{\partial Q^{(2)}}{\partial z_{ks}} = 2\sum_{j=1}^{J}\left(f_j^{'}(\bar{d}_{js})\left(\sum_{r=0}^{K} w_{jr}^{(2)} z_{rs} - \bar{d}_{js}\right)\right) f_j^{'}(\bar{d}_{js}) w_{jk}^{(2)}$$

being $\bar{z}_{ks} = g_k^{-1}(z_{ks})$, $\bar{d}_{js} = f^{-1}(d_{js})$, $k = 1, \ldots, K$, $j = 1, \ldots, J$ and $z_{0s} = 1, \forall s$

Step 5: Update intermediate outputs. Using the Taylor series approximation over the cost function, $Q(\mathbf{z} + \Delta\mathbf{z}) = Q(\mathbf{z}) + \sum_{k=0}^{K} \sum_{s=1}^{S} \frac{\partial Q(\mathbf{z})}{\partial z_{ks}} \Delta z_{ks} \approx 0$, the following increments are calculated to update the desired outputs of the hidden neurons

$$\Delta\mathbf{z} = -\rho \frac{Q(\mathbf{z})}{||\nabla Q||^2} \nabla Q, \quad \text{where } \rho \text{ is step size} \tag{4}$$

The procedure continues from Step 1 until a convergence condition is achieved.

The complexity of this method is determined by the complexity of Step 1 which solves systems of linear equations. Several efficient methods can be used with a complexity of $O(n^2)$, being n the number of weights of the network.

3 Experimental Results

In this section the performance of the proposed method and a comparative analysis with the original SBLLM is illustrated by its application to several regression and classification problems. These data sets were obtained from the Eric's Home page[1], the StatLib Datasets Archive[2], the Time Series Data Library[3], the UCI Machine Learning Repository[4] and the Data Mining Institute of the University of Wisconsin[5]. The conditions of the experiments are summarized in tables 1 and 2, regarding the number of samples of each data set, the topology of the neural network and the type of the cross-validation (10-fold cross-validation or leave-one-out) used to estimate the true error. Each experiment was run 10 simulations and the results are presented as the mean of these trials. Finally, to check the influence of the regularization parameter α, the proposed method was run with different $\alpha \in \{0, 0.002, 0.004, \ldots, 0.03\}$. The results obtained for all data sets are similar. As an example, fig. 2 shows the mean train and test error curve calculated over all simulations for each α for the Oscillation data set. Notice that the point $\alpha = 0$ corresponds to the method with no regularization term (the original SBLLM). As can be observed, although the train error increases with α the test error monotonically decreases down to a certain optimum α. After this point, the test error increases due to the excessive influence of the regularization term in the cost function.

In tables 3 and 4 is shown the performance of the method obtained with $\alpha = 0$ and using the optimum value of α. This performance is measured by mean squared error (MSE) in table 3 and the classification accuracy in table 4.

[1] http://www.cse.ogi.edu/∼ericwan/data.html
[2] http://lib.stat.cmu.edu/datasets
[3] http://www-personal.buseco.monash.edu.au/∼hyndman/TSDL
[4] http://www.ics.uci.edu/∼mlearn/MLRepository.html
[5] http://www.cs.wisc.edu/dmi/

Data Set	Samples	Topology	CV
Lorenz	500	8-10-1	10-Fold
Leuven	500	8-10-1	10-Fold
Dow-Jones	500	8-10-1	10-Fold
Waves	150	7-15-1	10-Fold
Henon	150	7-15-1	10-Fold
Kobe	100	5-10-1	10-Fold
TreeRings	110	5-12-1	10-Fold
CO2	40	8-15-1	Loo
Oscillation	20	4-15-1	Loo
Mackey-Glass	50	5-12-1	Loo
Blowfly	50	7-10-1	Loo

Table 1: Characteristics of the regression data sets.

Data Set	Samples	Topology	CV
Iris data	150	4-7-3	10-Fold
Lenses data	24	4-4-3	Loo
Forest CoverType	500	54-100-2	50,620 test samples
Dim Data	150	14-10-2	10-Fold
Housing Data	150	13-10-2	10-Fold

Table 2: Characteristics of the classification data sets.

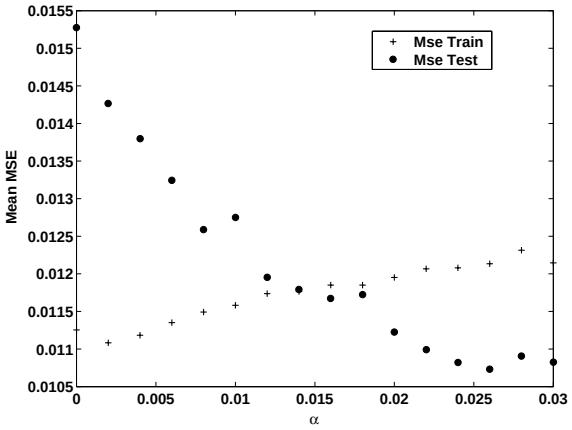

Fig. 2: Mean MSE curve as a function of α for the Oscillation data set.

As can be observed in all cases, the use of α improves the performance as was demonstrated by applying the Kruskal-Wallis test to check that the differences were statistically significant.

4 Conclusions

In this paper a generalized version of the Sensitivity-Based Linear Learning Method is presented by adding a regularization term. As was experimentally

Data Set	**Mse Train/Test($\alpha = 0$)**	**Mse Train/Test(optimum α)**
Lorenz	7.90×10^{-3} / 1.96×10^{-4}	8.20×10^{-3} / 3.88×10^{-5} (0.012)
Leuven	7.90×10^{-3} / 1.45×10^{-4}	8.10×10^{-3} / 3.89×10^{-5} (0.010)
Dow-Jones	7.90×10^{-3} / 1.45×10^{-4}	8.20×10^{-3} / 3.90×10^{-5} (0.010)
Waves	1.23×10^{-2} / 0.24×10^{-2}	1.29×10^{-2} / 0.20×10^{-2} (0.004)
Henon	2.61×10^{-2} / 5.50×10^{-3}	2.67×10^{-2} / 5.40×10^{-3} (0.030)
Kobe	1.22×10^{-2} / 1.40×10^{-3}	1.24×10^{-2} / 9.26×10^{-4} (0.004)
TreeRings	2.91×10^{-2} / 9.40×10^{-3}	3.00×10^{-2} / 8.10×10^{-3} (0.030)
CO2	1.35×10^{-2} / 1.98×10^{-4}	1.35×10^{-2} / 1.20×10^{-4} (0.002)
Oscillation	1.13×10^{-2} / 1.53×10^{-2}	1.21×10^{-2} / 1.07×10^{-2} (0.026)
Mackey	2.63×10^{-2}/ 2.40×10^{-3}	2.64×10^{-2} / 2.30×10^{-3} (0.020)
Blowfly	1.94×10^{-2} / 2.00×10^{-3}	1.96×10^{-2} / 1.90×10^{-3} (0.007)

Table 3: Mean Train and Test MSE values for $\alpha = 0$ and the optimum α

Data Set	**Acc. Train/Test($\alpha = 0$)**	**Acc. Train/Test(optimum α)**
Iris	84.90 / 82.53	94.15 / 92.67 (0.004)
Lenses	91.89 / 76.67	93.73 / 79.58 (0.030)
Forest	80.06 / 72.02	80.30 / 72.99 (0.100)
Dim Data	88.19 / 80.00	89.67 / 90.00 (0.010)
Housing	89.87 / 84.00	90.15 /89.00 (0.002)

Table 4: Mean Train and Test Accuracy values for $\alpha = 0$ and the optimum α

shown, this algorithm improves the performance of the original one when overfitting is possible. One of the distinctive features of the original SBLLM is that it measures the error before the nonlinearities. Regarding this, another enhancement of the algorithm was introduced by adding a scaling term to the loss function that improves the equivalence between the errors calculated before and after the nonlinearities.

References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Willian. Learning representations of back-propagation errors. *Nature*, 323:533–536, 1986.

[2] E. Castillo, B. Guijarro-Berdiñas, O. Fontenla-Romero, and A. Alonso-Betanzos. A very fast learning method for neural networks based on sensitivity analysis. *Journal of Machine Learning Research*, 7:1159–1182, 2006.

[3] G. E. Hinton. Learning translation invariant recognition in massively parallel networks. In *Volume I: Parallel architectures on PARLE: Parallel Architectures and Languages Europe*, pages 1–13, London, UK, 1987. Springer-Verlag.

[4] E. Castillo, O. Fontenla-Romero, A. Alonso Betanzos, and B. Guijarro-Berdiñas. A global optimum approach for one-layer neural networks. *Neural Comp.*, 14(6):1429–1449, 2002.

[5] D. Erdogmus, O. Fontenla-Romero, J.C. Principe, A. Alonso-Betanzos, and E. Castillo. Linear-least-squares initialization of multilayer perceptrons through backpropagation of the desired response. *IEEE Transactions on Neural Networks*, 16(2):325–337, 2005.